پایان‌نامه کارشناسی رشته فناوری اطلاعات

گرایش فناوری اطلاعات

موضوع:

۱۰۰ الگوریتم پرکاربرد در برنامه‌نویسی

استاد راهنما:

جناب آقای مهندس حمیدرضا نیرومند

نگارش:

صدیقه شهبازی

تابستان ۹۵

فهرست

تقدیر و تشکر

به مصداق «من لم یشکر المخلوق لم یشکر الخالق» بسی شایسته است از استاد؛

فرهیخته و فرزانه **جناب آقای مهندس حمیدرضا نیرومند** که با کرامتی چون خورشید، سرزمین دل را روشنی بخشیدند و گلشن سرای علم و دانش را با راهنمایی‌های کار ساز و سازنده بارور ساختند؛ تقدیر و تشکر نمایم.

همچنین از پدر و مادر عزیز، دلسوز و مهربانم که آرامش روحی و آسایش فکری فراهم نمودند تا با حمایت های همه جانبه در محیطی مطلوب، مراتب تحصیلی و نیز پایان‌نامه درسی را به نحو احسن به اتمام برسانم؛ سپاسگزاری نمایم.

(و یزکیهم و یعلمهم الکتاب و الحکمه)

صدیقه شهبازی

فصل اول

کلیات پژوهش

۱-۱- چکیده

هدف از هر برنامه‌ای پیاده‌سازی یک الگوریتم بر روی یک ماشین یا ابزار (همانند کامپیوتر، موبایل، قطعه‌ای الکترونیکی و...) و بهره‌گیری از منابع سخت‌افزاری و نرم‌افزاری آن می‌باشد.

برنامه شامل مجموعه‌ای از دستورات است که طی یک سیکل و ترتیب منظم و مشخص اجرا می‌شوند. این دستورات ابتدا در یک فایل متنی نوشته می‌شوند که به آن فایل منبع[1] می‌گویند سپس توسط یک مترجم[2] به زبان ماشین ترجمه (تبدیل) می‌شوند.

حاصل این ترجمه فایلی اجرایی[3] است که درون یک پلت فرم[4] مناسب اجرا می‌شود. این پلت فرم به عنوان مثال می‌تواند سیستم عامل ویندوز کامپیوترهای شخصی یا سیستم موبایل باشد.

هر کدام از دستورات تشکیل‌دهنده یک برنامه اصولاً ساده هستند و تنها یک عمل کوچک و مشخص را انجام می‌دهند. برنامه‌نویسان با کنار هم قرار دادن این دستورات (بر طبق یک الگوریتم مشخص) یک برنامه را خلق می‌کنند.

این برنامه می‌تواند کوچک و در حد چند دستور و یا بسیار بزرگ و پیچیده در حد چند صد هزار دستور باشد.

این کتاب شامل ۱۰۰ الگوریتم پرکاربرد برنامه‌نویسی است که اجرای یک الگوریتم را گام به گام به زبانی ساده‌تر بیان می‌کند.

[1] Source File
[2] Compiler
[3] Execut File
[4] Platform

۲-۱- انواع الگوریتم

الگوریتم‌های ریاضی

الگوریتم‌های بهینه‌سازی

الگوریتم‌های فرا ابتکاری

الگوریتم‌های ژنتیک

الگوریتم‌های هوش مصنوعی

الگوریتم‌های جستجوگر

الگوریتم‌های مرتب‌سازی

الگوریتم‌های فازی

الگوریتم‌های تکاملی و...

۳-۱- مقدمه

الگوریتم (نسبت: الگوریتمی، خوارزمیک) یا خوارزمی مجموعه‌ای متناهی از دستورالعمل‌ها است، که به ترتیب خاصی اجرا می‌شوند و مسئله‌ای را حل می‌کنند. به عبارت دیگر یک الگوریتم، روشی گام به گام برای حل مسئله است. شیوه محاسبه معدل در مدرسه، یکی از نمونه‌های الگوریتم است.

تمام الگوریتم‌ها باید شرایط و معیارهای زیر را دارا باشند[1] :

- ورودی :

یک الگوریتم باید هیچ یا چندین پارامتر را به عنوان ورودی بپذیرد؛

- خروجی:

الگوریتم بایستی حداقل یک کمیت به عنوان خروجی (نتیجه عملیات) تولید کند؛ [1]

- قطعیت:

دستورهای الگوریتم باید با زبانی دقیق، و بی‌ابهام بیان شوند. هر دستورالعمل نیز باید انجام‌پذیر باشد. دستورهایی نظیر «مقدار ۶ یا ۷ را به x اضافه کنید» یا «حاصل تقسیم پنج بر صفر را محاسبه کنید» مجاز نیستند؛ چرا که در مورد مثال اول، معلوم نیست که بالاخره چه عددی باید انتخاب شود، و در خصوص مثال دوم هم تقسیم بر صفر در ریاضیات تعریف نشده‌است. [1]

- محدودیت:

الگوریتم باید دارای شروع و پایان مشخصی باشد، به نحوی که اگر دستورهای آن را دنبال کنیم، برای تمامی‌حالات، الگوریتم پس از طی مراحل شمارا و متناهی خاتمه یابد. به علاوه، زمان لازم برای خاتمه الگوریتم هم باید به گونه‌ای معقول، کوتاه باشد. [1]

در علوم رایانه، یک الگوریتم را یک روال محاسباتی خوش‌تعریف می‌دانند، که مقدار یا مجموعه‌ای از مقادیر را به عنوان ورودی[5] دریافت کرده و پس از طی چند گام محاسباتی، ورودی را به خروجی[6] تبدیل می‌کند. بجز این، الگوریتم را ابزاری برای حل مسائل محاسباتی نیز تعریف کرده‌اند. ساخت و طراحی الگوریتم مناسب در مرکز فعالیت‌های برنامه‌سازی رایانه قرار دارد. یک برنامه رایانه‌ای، بیان یک یا چند الگوریتم با یک زبان برنامه‌نویسی است. [1]

- مفهوم:

[5] Input
[6] Output

مفهوم الگوریتم را معمولاً با تشبیه به دستور آشپزی توضیح می‌دهند. مثلاً اگر بخواهیم آبگوشت درست کنیم (عمل مورد نظر) با فرض اینکه مواد خام را داریم (حالت اولیه) مراحل مشخصی را باید طبق دستور آشپزی طی کنیم (دستورالعمل‌ها) تا به آبگوشت آماده (حالت پایانی) برسیم. البته الگوریتم‌ها معمولاً پیچیده‌تر از این هستند.

الگوریتم گاه دارای مراحلی است که تکرار می‌شود (در مثال آبگوشت مثلاً چند بار باید نمک زد یا آب اضافه کرد) و یا در مرحله‌ای نیازمند تصمیم‌گیری است (اگر نمک کافی است دیگر نمک نمی‌زنیم، اگر کافی نیست نمک می‌زنیم).

اگر الگوریتم برای عمل مورد نظر مناسب نباشد و یا غلط باشد به نتیجه مورد نظر نمی‌رسیم. مثلاً اگر الگوریتم آبگوشت را با مواد اولیه کباب انجام دهیم واضح است که به آبگوشت نمی‌رسیم.

باید بدانیم برای هر الگوریتم تعریف متغیرها و طراحی مرحله به مرحله بسیار مهم است. زیرا الگوریتم باید بداند بر روی چه متغیرهایی، چه اعمالی را انجام دهد و نتیجه را در غالب چه متغیرها یا پارامترهایی نشان دهد. [1]

فصل دوم

۱۰۰ الگوریتم پرکاربرد در برنامه‌نویسی

۲-۱- الگوریتم جمع و تفریق

1-1 برای اینکار ابتدا بعد از دریافت ورودی یک متغییر برای جمع‌های بیشتر از یک رقم تعریف کرد. آیا عدد ورودی تک رقمی‌است یا خیر اگر تک رقمی بود (a+b) می‌کنیم در غیر اینصورت یکان دهگان را جداگانه جمع می‌کنیم اگر جمع یکان عدد بیشتر از ده بود یکان را نگه می‌داریم و سپس در هرمرحله مقدار متغییر کمکی در مرحله قبل را در دهگان عدد جمع می‌کنیم.

1-2 برای اینکه بتونیم این مسئله رو حل کنیم ابتدا بعد از دریافت ورودی باید ببینیم کدام بزرگتراست. اگه عدد دوم (b) بزرگتر از عدد اول (a) بود باید جای آنها را تعوض کنیم. بعد عدد اول (a) را از عدد دوم (b) کم کرده. سپس نتیجه را نمایش بدهیم. [1]

۲-۲- الگوریتم ضرب و تقسیم

2-1 الگوریتم این برنامه خیلی ساده هست. به طوری‌که ابتدا مثبت یا منفی بودن عددهای ورودی رو با یه شرط مشخص می‌کنیم. بعد هم داخل یک حلقه که به تعداد عدد اول (a) تکرار میشه عدد دو (b) رو داخل یک متغیر قرار می‌دهیم و هر بار به اندازه عدد دوم به اون اضافه یا کم (بسته به مثبت یا منفی بودن اعداد) می‌کنیم. [1]

2-2 برای حل این مسئله ابتدا ورودی‌ها رو دریافت می‌کنیم. بعد چهار تا حالت پیش میاد که به صورت زیر هست:

2-2-1 هر دو عدد مثبت باشن.

2-2-2 هر دو عدد منفی باشن.

2-2-3 عدد اول منفی و عدد دوم مثبت باشه.

2-2-4 عدد اول مثبت و عدد دوم منفی باشه.

بعد متناسب با حالتی که پیش میاد اعداد رو داخل حلقه‌ها کم یا زیاد می‌کنیم تا جواب به دست بیاد. [1]

۲-۳- الگوریتم توان

الگوریتم این برنامه آسون هست. ابتدا ورودی‌هایمان را دریافت می‌کنیم. بعد به تعداد عدد دوم(b)، عدد اول (a) را به خودش ضرب می‌کنیم و در یک متغیر قرار می‌دهیم و برای این کار از حلقه استفاده می‌کنیم.

الگوریتم این مسئله خیلی شبیه الگوریتم مسئله ضرب هست. [1]

عددی را از ورودی دریافت می‌کنیم سپس A*A قرار داده و پاسخ آن را در خروجی می‌آوریم.

۴-۲- الگوریتم حاصل A را به توانB

دوعدد از ورودی دریافت کرده و یک عدد برای جمع گذاشته می‌شود. سپس حلقه‌ای تعریف می‌کنیم که از i=0 باشد تا انتهای عدد ۲ و مساوی عدد دوم بود تکرار کن C=C+A و اگر C مساوی با عدد دوم بود به پایان برود و C را چاپ کنید. [1]

برای حل این الگوریتم دو متغییر به عنوان پایه و نما توان مانند x و y در نظر گرفته‌ام. حال اگر بخواهیم عمل توان را پیاده‌سازی نماییم احتیاج است از خاصیت عضو خنثی در ضرب استفاده شود. همانطور که می‌دانید عضو خنثی در ضرب، عدد ۱ می‌باشد و نیاز به یک حلقه داریم تا عدد دوم باید شمارش شود و مقدار متغییر عضو خنثی در داخل حلقه ضرب در عدد اول می‌شود تا در اتمام کار حلقه توان محاسبه شود.

۱. شروع
۲. دو متغییر x و y را بخوان
۳. عدد ۱ را به متغییر p نسبت بده
۴. عدد ۱ را به متغییر i نسبت بده
۵. اگر مقدار متغییر i کوچکتر از متغییر y باشد برو به مرحله ۷
۶. در غیر این صورت برو به مرحله ۹
۷. متغییر p را ضرب در متغییر x بکن و حاصل را در متغییر p بریز
۸. یک واحد به متغییر i اضافه کن
۹. برو به مرحله ۵
۱۰. مقدار متغییر p را چاپ کن
۱۱. پایان

۵-۲- الگوریتم بنویسید ده عدد از کاربر و معدل آن را محاسبه کند

توضیح: ۱۰ عدد را از ورودی دریافت می‌کنیم و سپس حلقه‌ای تعریف می‌کنیم که ده بار عدد دریافتی را با عدد جدید در یک متغییر جمع می‌کنیم (a=a+b) آنگاه اگر ۱۰عدد دریافت شد متغییر جمع را در ۱۰ تقسیم کنیم و درخروجی را پاسخ تقسیم را چاپ می‌کنیم. در غیر اینصورت حلقه را تکرار می‌کنیم تا به ۱۰ بار رسیده شود و از حلقه خارج گردد. [1]

۶-۲- الگوریتم محاسبه‌ی فاکتوریلa

1) شروع
2) عدد را قرار بده درa
3) یک را قرار بده در (n=1)
4) nرا در a ضرب کرده و دوباره در n قرار بده (n=n*a)
5) از a یکی کم کرده دوباره در a قرار دهید
6) اگر a بزرگتر از یک است برو به سطر ٤
7) عدد n را نشان بده
8) پایان

a!=(a) (a-1) (a-2) (a-3)…(3) (2) (1)

از دید کامپیوتری همان طور که مشاهده می‌کنیم برای محاسبه‌ی فاکتوریل ما یک متغیر داریم که از آن یک واحد کم شده و در عین حال در اعداد قبلی ضرب می‌شود در برنامه‌ی بالا متغیر مورد نظر n بوده و برای سیر نزولی اعداد از خود متغیر a استفاده کردیم. با اجرای خط ۴ و ۵ همزمان هم a در متغیر مورد نظر ضرب شده هم از آن یک واحد کم می‌شود در خط ۶ یک کنترل گذاشته شده تا a به صفر (۰) نرسد چون اگر a برابر یک (۱) شود و دوباره حلقه ادامه پیدا کند در این دور a صفر شده و n را صفر کرده و تمام محاسبات خراب می‌شود. در آخر هم n عدد فاکتوریل ما است. [3]

توضیح: عدد ورودی دریافت می‌کنیم و داخل یک متغییر می‌گذاریم (a) سپس متغییری برای نگه داری جمع تعریف می‌کنیم و مقدار ۱ را اختصاص می‌دهیم (n=1) و تاجایی که عدد ورودی بزرگتر از یک بود مقدار متغییر را در عدد ورودی ضرب و در هر مرحله یک واحد از عدد ورودی کم می‌کنیم و دوباره در متغییر n ضرب می‌نماییم و تا آخر به عددی رسیدیم و شرط پایان پذیرفت جواب را چاپ می‌کنیم. [3]

۷-۲- الگوریتمی بنویسید که N عدد صحیح دریافت کرده بزرگ‌ترین و کوچک‌ترین عدد را یافته و چاپ کند

1) شروع
2) A را دریافت کن
3) N را دریافت کن

4) A ← max

5) A← min

6) c ← 2

7) مادامی c < = n تکرار کن

{

8) A را دریافت کن

9) اگر a > max آنگاه max ← a

10) اگر a < min آنگاه min ← a

11) c ← C+1

}

12) Max و min را چاپ کن

13) پایان

۸-۲- کوچک‌ترین عدد در ۱۰۰۰ عدد دریافتی

توضیح: فرض کنید که به شما لیستی از اعداد را می‌دهند، برای پیدا کردن کوچکترین عدد در لیست اولین عدد را به عنوان کوچکترین در نظر می‌گیرید سپس عدد بعدی را با آن مقایسه می‌کنیم، اگر عدد جدید از عدد قبلی کوچکتر بود عدد جدید را به عنوان کوچکترین در نظر می‌گیرید و گر نه همان عدد قبلی کوچکترین خواهد بود. این روند را تا انتهای لیست ادامه می‌دهید؛ در پایان عددی که در هر بررسی به عنوان کوچکترین عدد بود، جواب مورد نظر ما خواهد بود. توجه کنید که در این روال شما همواره یک عدد را در ذهن خود در نظر گرفته بودید، برای نوشتن الگوریتم مورد نظر ما یک خانه حافظه را به کوچکترین عدد درهر مرحله اختصاص می‌دهیم. [3]

۱) min را دریافت کن.

2) i =1 -2

3) aرا دریافت کن

4) اگر a<min آنگاه min = a.

5) i = i + 1.

6) اگر i>=1000 به مرحله ۸ برو.

7) به مرحله ۳ برو.

8) min را چاپ کن.

۹-۲- الگوریتم مجموع اعداد طبیعی مضرب ۳ و کوچکتر از ۵۰ است

توضیح: یک متغییر برای جمع تعریف می‌کنیم (sum=0) و یک متغییر شرط با مقدار ۳ قرار می‌دهیم I=3 تا جایی که sum=sum+I را در هر مرحله متغییر شرطی با متغییر جمع می‌کنیم. در هربار متغییر شرطی را با ۳ جمع می‌کنیم. اگر متغییر شرطی کوچکتر از ۵۰ بود و در پایان متغییر جمع را چاپ می‌کنیم. [2]

۱۰-۲- الگوریتمی بنویسید که یک عدد به واحد میلی‌متر دریافت کرده، معین کند چندمتر و چند سانتی متر و چند میلی متر است

هر یک متر ۱۰۰ سانتی متر و ۱۰۰۰ میلی متر است. هر سانتی متر هم ۱۰ میلی متر محسوب می‌شود

توضیح: عدد ورودی به میلی متر است پس از دریافت آن را در ۱۰۰۰ تقسیم میکنیم تا متر بدست اید سپس باقی‌مانده‌اند را در متغییری می‌ریزیم و در ۱۰۰ تقسیم می‌کنیم سانتی‌متر به دست می‌آید. و در انتها باقیمانده تقسیم بر ۱۰۰ را میلی‌متر قرار می‌دهیم. [4]

۱) شروع

۲) mmرا دریافت کن

۳) m= N \ 1000

۴) mm ← N mod 1000

۵) cm ← mm \ 100

۶) mm ← Mm mod 100

۷) mm , cm , m را چاپ کن

۸) پایان

۱۱-۲- الگوریتم ۳ عدد از ورودی دریافت آیا اضلاع یک مثلث می‌باشد

توضیح: سه عدد از ورودی دریافت می‌کنیم و شرط می‌گذاریم اگر جمع دوعدد اول بزرگتر عدد سوم و عدد دوم از عدد اول بزرگتر و جمع عدد اول و عدد سوم بزرگتر از عدد سوم بود چاپ کن اضلاع مثلث است.

۱۲-۲- الگوریتمی بنویسید که طول و عرض یک مستطیل را دریافت کرده، محیط و مساحت مستطیل را محاسبه کند [3]

توضیح: فرمول محیط مستطیل میشه طول + عرض رو ضرب در ۲ می‌کنیم و همچنین مساحت رو طول در عرض ضرب می‌کنیم.

دو عدد یکی به عنوان عرض و دیگی به عنوان طول وارد می‌کنیم و همانند فرمول عمل می‌کنیم و سپس عدد به دست امده رو چاپ می‌کنیم. [3]

۱) شروع

۲) x , yرا دریافت کن

۳) x * y ← masahat

۴) (x+y)*2 ← mohit

۵) mohitو masahatرا چاپ کن

۶) پایان

۱۳-۲- الگوریتمی‌بنویسید که شعاع دایره را دریافت کند، مساحت و محیط دایره را بدست آورد

توضیح: مساحت برابر شعاع ضرب در شعاع جمع آن با ۳,۱۴ و محیط دایره برابر ا ست با ضرب شعاع در ۳,۱۴ در عدد ۲

۱) شروع

۲) Rرا دریافت کن

۳) 3.14 *r*r ← masahat

۴) 2* 3.14 *r ← mohit

۵) masahatو mohitرا نمایش بده

۶) پایان

۲-۱۴- جابجایی مقادیر دو عدد بدون استفاده از متغیر کمکی

توضیح: ما ابتدا یک آرایه تعریف کردیم با یک متغیر به نام temp که بتوانیم با استفاده از آن مقادیر آرایه را جا به جا کنیم سپس در for تودرتو الگوریتم مرتب‌سازی را اجرا کردیم و در for بعدی آرایه‌ی مرتب شده را نمایش دادیم. [4]

شرط ما در if این است که اگر مقدار اندیس آرایه با مقدار اندیس‌های بعد از خود بزرگتر بود آن‌ها را با متغیر کمکی جابجا کن.

برای اینکه ما دو مقدار از دو اندیس آرایه را با هم جابه جا کنیم به یک متغیر کمکی نیاز داریم و به همان صورت که در داخل بلوک if می‌بینید دو مقدار را جابه جا می‌کنیم. [4]

در این شرط آرایه ما آرایه را به صورت صعودی مرتب کردیم ولی اگر شما بخواهید آن را به صورت نزولی مرتب کنید فقط کافیست شرط آرایه را بر عکس کنید به صورت:

برای جا به جایی دو مقدار در دو متغیر ما به متغیر کمکی نیاز داریم ولی اگر کسی بخواهد بدون استفاده از آن متغیر کمکی دو مقدار را جا به جا کند باید چه کار کرد؟

ما می‌توانیم با استفاده از عملگرهای محاسباتی آن دو مقدار را جا به جا کنیم. [4]

۱) a,b را دریافت کن

۲) a+b ← a

۳) a-b ← b

۴) a-b ← a

۵) پایان

۲-۱۵- الگوریتمی بنویسید که دو متغیر عددی A,B را دریافت کرده، سپس محتوای آنها را با هم تعویض کند [4]

توضیح: ابتدا یک متغییر کمکی قرار می‌دهیم تا بتوانیم مقادیر را جابجا کنیم. آنگاه دو عدد ورودی را گرفته (a,b) مقادیر a را درون متغییر کمکی گذاشته سپس مقدار b را درون a می‌ریزیم و آنگاه با خالی شدن b مقدار a را از متغییر کمکی درون b قرار می‌دهیم.

۱) شروع

۲) A, B را دریافت کن

۳) 0 ← temp

۴) A ← temp

۵) B ← A

۶) temp ← B

۷) پایان

۲-۱۶- الگوریتمی‌بنویسید که عددی را دریافت کرده و معین کند زوج است یا فرد

توضیح: عددی را از ورودی دریافت می‌کنیم شرط می‌گذاریم که اگر باقی مانده ی تقسیم عدد ورودی بر ۲ برابر با صفر بود انگاه زوج است در غیر اینصورت نیست[4]

۱) شروع

۲) Aرا دریافت کن

۳) اگر a mod 2 = 0 آنگاه

a. پیغام a " زوج است " را نمایش بده

۴) در غیر اینصورت

a. پیغام a" فرد است " را نمایش بده

۵) پایان

۲-۱۷- دو عدد دریافت اعداد بین آن را چاپ کند

توضیح: دوعدد را از ورودی دریافت می‌کنیم (A, B) و اگر عدد اول بزرگتر از عدد دوم بود. عدد دوم را چاپ کن در غیر اینصورت عدد اول را چاپ کن و به عدد اول یک واحد اضافه کن اگر عدد اول کوچکتر از عدد دوم بود عدد اول را چاپ کن در غیر اینصورت عدد دوم را چاپ کن و به عدد دوم یک واحد اضافه کن اگر عدد دوم کوچکتر یا مساوی عدد دوم بود عدد را چاپ کن[4] .

۱۸-۲- الگوریتمی‌بنویسید که اعدادA و Bرا دریافت کرده اعداد فرد بینA و B را چاپ کند

۱) شروع

۲) a , b را دریافت کن

۳) اگر a > b آنگاه

۴) a ← max

۵) b ← min

۶) در غیر اینصورت

۷) b ← max

۸) a ← min

۹) اگر min mod 2 =0 آنگاه min ← min+1 در غیر این صورت min +2 ← min

۱۰) مادامی‌که min < = max تکرار کن

{

i. minرا چاپ کن

a. min +2 ← min

}

۱۱) پایان

۱۹-۲- الگوریتمی‌بنویسید که شماره یک روز و شماره یک ماه از سال را دریافت کرده، معین کند چند روز از سال می‌گذرد [2]

(مثال: اگر تاریخ ۳/۲ وارد شد عدد ۶۴ نمایش داده شود.)

۱) شروع

۲) d , mرا دریافت کن

۳) اگر (m-1) < =6 آنگاه

i. `31 +d (m-1) ← days`

در غیر اینصورت

i. `(m-7)*30 +(6*31) +d ← days`

۴) `days`را چاپ کن

۵) پایان

۲۰-۲- الگوریتمی‌بنویسید که عددی را دریافت کرده مقسوم علیه‌های آن را چاپ کند

توضیح: برای حل این الگوریتم احتیاج به یک حلقه می‌باشد این حلقه به تعداد دفعات عدد دریافتی باید تکرار شود، در هر مرتبه که تکرار می‌شود اگر باقیمانده عدد دریافتی از ورودی بر شمارنده حلقه برابر ۰ باشد شمارنده چاپ خواهد شد که شمارنده در حکم مقسوم علیه خواهد بود N عدد طبیعی می‌باشد. مقسوم علیه N، اعداد صحیح کوچکتر از N است اگر N به هرکدام از آنها تقسیم شود باقیمانده صفر می‌شود.ب رای مثال عدد ۱۲ را در نظر بگیریم که مقسوم علیه آن ۱و۲و۳و۴و۶و۱۲ است. و در اینجا یک عدد دریافت می‌کنیم متغییر برای شمارش گذاشته که هر بار عدد باقی در آن تقسیم شود و یکی ازآن کم گردد اگر باقی مانده صفر شد مرحله یک عدد بر متغییر شمارش اضافه کند اگر کوچکتر از عدد ورودی بود دوباره محاسبه نماید. [2]

۱) شروع

۲) `n`را دریافت کن

۳) `c ← 1`

۴) مادامی‌که `c < = n` تکرار کن

{

i. اگر `n mod c =0` آنگاه `c` را چاپ کن

i. `c ← c+1`

}

۵) پایان

۲۱-۲- الگوریتم تعمیم‌یافته اقلیدس

توضیح: در علم حساب و برنامه‌نویسی کامپیوتر الگوریتم تعمیم یافته اقلیدس تعمیمی‌بر الگوریتم اقلیدس است که در کنار محاسبه بزرگترین مقسوم علیه مشترک دو عدد صحیح a,b ضرایب قضیه بزو را هم محاسبه می‌کند(x , y اعداد صحیح اند):

a x + b y = gcd (a , b)

همچنین می‌توان بدون هزینه اضافی خارج قسمت a , b را با بزرگترین مقسوم علیه مشترک محاسبه کرد. [2]

الگوریتم تعمیم یافته اقلیدس به الگوریتمی‌بسیار مشابه به الگوریتمی‌برای محاسبه بزرگترین مقسوم علیه مشترک چندجمله ای و ضرایب قضیه بزو از دو چندجمله ای با یک متغیر، ارجاع دارد. الگوریتم تعمیم یافته اقلیدس زمانی که a و b نسبت به هم اولند مفیدتر است. چون x برابر وارون ضربی پیمانه‌ای a به پیمانه b و y برابر وارون ضربی پیمانه‌ای b به پیمانه a است. در الگوریتم اقلیدس برای چند جمله‌ای می‌توان وارون ضربی را در بسط‌های میدان جبری محاسبه کرد. [3]

که از آن نتیجه می‌شود هردو نوع الگوریتم اقلیدسی تعمیم یافته (معمولی و چند جمله‌ای) کاربردی گسترده در رمزنگاری دارند. به خصوص در محاسبه وارون ضربی پیمانه‌ای در روش RSA یک گام ضروریست. [3]

توضیح: در الگوریتم اقلیدسی معمولی فقط باقی‌مانده‌ها نگاه داشته می‌شوند و خارج قسمت استفاده نمی‌شود. اما در الگوریتم تعمیم یافته خارج قسمت‌های متوالی استفاده می‌شوند. به طور دقیق‌تر در الگوریتم تعمیم یافته که a، b را به عنوان ورودی می‌گیرد، دنباله q1…..q(k) از خارج قسمت‌ها و دنباله r0.....r(k)+1 از باقی‌مانده‌ها را شامل میشود. [3]

مهمترین ویژگی تقسیم اقلیدسی نامساوی سمت راست است که r i + 1 را متفاوت از r i − 1 و ri تعریف می‌کند. وقتی باقی‌مانده به صفر رسید محاسبه متوقف می‌شود، بزرگترین مقسوم علیه مشترک آخرین باقی‌مانده غیر صفر است. الگوریتم تعمیم یافته اقلیدسی روشی مشابه دارد با این تفاوت که دو دنباله زیر نیز به آن اضافه می‌شود: [2]

در این الگوریتم هم وقتی r k + 1 = 0 میشود کار الگوریتم به اتمام می‌رسد.

r k بزرگ‌ترین مقسوم علیه برای دو ورودی a=R0 و b=R1 است.

ضرایب بزو هم که برابر sk و tk , در رابطه زیر صدق می‌کنند:

خارج قسمت تقسیم a و b بر بزرگترین مقسوم علیه مشترک‌شان به این صورت است:

که این یعنی جفت ضرایب بزو که از این الگوریتم به دست می‌آیند یکی از کمترین جفت ضرایب بزو هستند. [2]

۲۲-۲- الگوریتم عدد کامل

توضیح: عدد کامل عددی است که مجموع مقسوم‌علیه‌های آن عدد (به جز خود عدد) برابر با خود عدد باشد.

مثال:

مقسوم علیه‌های عدد ۶ برابر است با ۱ و ۲ و ۳ که جمع این سه عدد برابر است با عدد ۶ یعنی عدد ۶ یک عدد کامل است.

پس ما برای این کار باید از یک `for` استفاده کنیم که متغیر شمارنده‌ی آن از شماره‌ی ۱ شروع شود و تا یکی مانده به عدد تمام شود. در داخل بلوک کد `for` باید بنویسیم که اگر باقی‌مانده‌ی تقسیم `a` بر `i` برابر با صفر باشد(یعنی یکی از مقسوم علیه های عدد`a`) بعلاوه ی متغیر `s` می‌شود و در `s` قرار می‌گیرد. همین عمل چند بار انجام می‌گیرد. [3]

در آخر باید برسی کنیم که عدد `s` برابر با عدد `a` هست یا خیر. اگر برابر باشد عدد کامل است واگرنه عدد کامل نیست. [3]

خروجی اول:

```
6
adad kamel ast
```

خروجی دوم:

```
5
adad kamel nist
```

۲۳-۲- الگوریتمی‌بنویسید که عددی را دریافت کرده و معین کند اول است یا خیر

توضیح: عدد اول عددی طبیعی بزرگ‌تر از ۱ است که بر هیچ عددی بجز خود و ۱ بخش‌پذیر نباشد. تنها استثنا عدد ۱ است که جزو این اعداد قرار نمی‌گیرد. اگرعددی طبیعی وبزرگ‌تر از ۱ اول نباشد مرکب است. [3]

رقم یکان اعداد اول بزرگ‌تر از ۱۰ فقط ممکن است ارقام ۱، ۳، ۷، و ۹ باشد.

عددی را از ورودی دریافت می‌کنیم و یک متغییر را ۲ قرار می‌دهیم (X=2) قرار می‌دهیم و شرط می‌گذاریم از ۱ تا خود عدد تکرار کند و یک یک شمارنده را معادل یک قرار می‌دهیم W=1 .اگر باقی مانده تقسیم آن صفر شود عدد اول است در غیر اینصورت عدد اول نیست. [3]

- خواص اعداد اول:

هر عدد اول برابر است با ٦ n+1یا ٦ n-1که n یک عدد صحیح است.

مجذور هر عدد اول برابر است با ٢٤ . n+1

تفاضل مجذورهای دو عدد اول مضربی از ۲۴ است.

حاصل ضرب هر دو عدد اول بجز ۲و۳ مضربی از ۶ بعلاوه یا من‌های یک است.

توان چهارم هر عدد اول بجز ۲و۳ مضربی از ۲۴۰ بعلاوه یک است. [3]

۱) شروع

۲) Nرا دریافت کن

۳) c ← 2و sum۰ ←

۴) مادامی‌که 2 \ n => c تکرار کن

{

۵) اگر c =0 n mod آنگاه sum ← sum +1

}

۶) اگر sum =0 آنگاه

a. پیغام " عدد اول است " را نمایش بده

۷) در غیر اینصورت

a. پیغام " عدد اول نیست " را نمایش بده

۸) پایان

۲-۲۴- الگوریتمی‌بنویسید که دو عدد را دریافت کرده، کوچکترین مضرب مشترک و بزرگترین مقسوم علیه مشترک آنها را چاپ کند

توضیح: از روش غربال برای ب م م به این ترتیب عدد بزرگتر را بر عدد کوچک تقسیم و اگر باقیمانده صفر شد عدد کوچکتر و که مقسوم علیه تقسیم است ب م م می‌باشد. و اگر صفر نشد مقسوم علیه قرار داده و دوباره عمل تقسیم انجام می‌دهیم تا زمانی که با قی مانده صفر شد که آنرا صفر باشد ب – م – م است. [3]

۱) شروع

۲) a,bرا دریافت کن

a. temp ←

۳) a*b ← mul

۴) مادامی‌که a mod b <> 0 تکرار کن

{

i. Temp←b

ii. a mod b ← b

iii. Temp ← a

}

۵) b ← bmm

۶) mul \ bmm ← kmm

۷) Bmmو kmm را نمایش بده

۸) پایان

۲-۲۵- الگوریتمی‌بنویسید که عدد A را دریافت کرده تعداد ارقام آن را چاپ کند

۱) شروع

۲) Nرا دریافت کن

۳) sum ← 0

۴) مادامی‌که n > 0 تکرار کن

{

i. n \ 10 ← n

ii. sum + 1 ← sum

}

۵) Sum را نمایش بده

۶) پایان

۲-۲۶- الگوریتمی‌بنویسید که عدد A را دریافت کرده مجموع ارقام آن را چاپ کند

توضیح: عدد مورد نظر T رقمی‌است و رقم یکان آن از تقسیم عدد بر ۱۰ بدست می‌آید که همان باقی مانده تقسیم است برای محاسبه رقم دهگان خارج قسمت تقسیم اولی (که در اصل عدد بر صحیح تقسیم عدد بر ۱۰ است)را دوباره بر ۱۰ تقسیم می‌کنیم با قیمانده آن رقم دهگان است و این تا زمانی انجام می‌شود.که خارج قسمت صفر شود. [3]

۱) شروع

۲) Nرا دریافت کن

۳) sum ← 0

۴) مادامی‌که n > 0 تکرار کن

{

i. n mod 10 ← b

ii. n\10 ← n

iii. sum + b ← sum

}

۵) Sumرا نمایش بده

۶) پایان

۲-۲۷- الگوریتمی‌بنویسید که عدد A را دریافت کرده معکوس آن را چاپ کند

مثال: ۳۸۱ === ۱۸۳

توضیح: ابتدا تعداد ارقام را محاسبه می‌کنیم. سپس حلقه‌ای تشکیل داده و به اندازه تعداد ارقام عدد هر بار رقم جداکنیم و سپس آن رقم را با توجه به محل قرارگیری‌اش در عدد اصلی، به صورت معکوس در عدد وارون قرار می‌دهیم. برای جداکردن ارقام از خارج قسمت عدد بر توان‌های عدد ۱۰ استفاده کنید. برای قراردادن ارقام در عدد وارون هر رقم را در توان از ۱۰ ضرب می‌کنیم و با عدد قبلی جمع می‌کنیم به گونه ای که عدد تشکیل شده وارون عدد گردد. [3]

۱) شروع

۲) Nرا دریافت کن

۳) sum ← 0

۴) مادامی‌که n > 0 تکرار کن

{

i. n mod 10 ← b

ii. n\10 ← n

iii. Sum*10 + b ← sum

}

۵) Sumرا نمایش بده

۶) پایان

۲۸-۲- تعیین علامت عدد ورودی

توضیح: یک عدد را از ورودی می‌خوانیم و در این صورت ۳ حالت پیش می‌آید که اگر A=0 عدد ورودی برابر با صفر صفر بود عدد صفر را چاپ کند. اگر عدد ورودی از صفر بزرگتر بود علامت عدد مثبت بوده و اگر عدد کوچکتر از صفر بود در خروجی چاپ کند علامت منفی خواهد بود. [3]

۲۹-۲- الگوریتمی‌بنویسید که عددی را از ورودی گرفته و قدر مطلق آن را حساب کند

توضیح: که هر عددی که درون قدرمطلق قرار بگیرد در خروجی مثبت می‌شود. قدر مطلق روی اعداد مثبت تاثیری ندارد چون آنها از اول مثبت هست پس اگر عددی مثبت داخل قدرمطلق قرار گرفت خروجی خود عدد خواهد بود .

عدد A را از ورودی دریافت می‌کنیم با استفاده از دستورالعمل شرطی، مثبت بودن عدد بررسی می‌شود اگر مثبت باشد قدر مطلق آن عدد برابر با خودش خواهد بود و عدد خروجی چاپ می‌کنیم در غیر اینصورت آن را در منفی یک (۱-) ضرب می‌کنیم تا مثبت شود، سپس عدد را در خروجی چاپ می‌کنیم.

۱) شروع

۲) nرا بگیر

۳) اگر n>=0 باشد برو به گام ۴

۴) در غیر اینصورت (n) * (-1)

۵) چاپ کن n را

۶) پایان

۳۰-۲- تعیین مربع کامل عدد ورودی

توضیح: مربع کامل عددی است که بتواند به صورت ضرب یک عدد صحیح در خودش نوشته شود. مثلاً ۳۶ برابر با ۶×۶ است و ۴۹ برابر با ۷×۷ است. [3]

تمام مربع‌های کامل به ۰، ۱، ۴، ۵، ۶ یا ۹ ختم می‌شود:

... و ۰، ۱، ۴، ۹، ۱۶، ۲۵، ۳۶، ۴۹، ۶۴، ۸۱، ۱۰۰، ۱۲۱،۱۴۴، ۱۶۹

۱) عدد را از ورودی گرفته و داخل متغییر از نوع صحیح قرار می‌دهیم

۲) جذر تابع[7]عدد رو گرفته و داخل متغییر از نوع اعشاری قرار داده

۳) عدد رو در خودش ضرب کرده و سپس حاصل ضرب را درون متغییر نگهداری می‌کنیم

۴) اگر حاصل برابر با عدد صحیح ورودی باشد یعنی عدد مربع کامل هست در غیر اینصورت مربع نیست

۵) چون متغییر از نوع اعشاری با دقت بالاست در نتیجه عددی که مربع نباشد رو گرد نمی‌کند و برنامه درست کار می‌کند.

[7] sqrt

۳۱-۲- رشته معکوس

*

* *

***

توضیح: یک عدد را برای تعداد خط‌های چاپ ستاره وارد می‌کنیم سپس تا جایی که (تعداد خط - سطر) I=1 کوچکتر از عدد وارد شده بود و همچنین ستون‌های آن از J=1 تا جایی که کوچکتر یا مساوی با مقدار I(خط) چاپ کن * و این عمل تکرار شود تا زمانی که به عدد وارد شده رسیدیم و پایان یابد.

۳۲-۲- الگوریتم دو معادله دو مجهولی

توضیح: ما می‌دانیم که برای حل یک دستگاه دو معادله‌ی دو مجهولی ابتدا باید یا x و یا y را از دو عبارت موجود حذف کنیم. البته در این سورس x از هر دو عبارت حذف می‌شود .

برای حذف x از هر دو عبارت باید ضریب x در هر دو عبارت صفر شود که برای اینکار، عبارت اول را بر قرینه‌ی عدد d ضرب می‌کنیم و عبارت دوم را نیز بر عدد a ضرب می‌کنیم.

بعد از انجام این کار یک دستگاه دو معادله ی دو مجهولی جدید به وجود می‌آید که در آن هر دو عبارت را با هم جمع می‌کنیم. در نتیجه x از عبارت‌ها پاک می‌شود اکنون ما با توجه به عبارت زیر y را بدست می‌آوریم:

```
by=c => y=c/b
```

بعد از به دست آوردن y آنرا در یکی از عبارت‌ها جایگزین می‌کنیم و x را با استفاده از فرمول زیر محاسبه می‌کنیم:

```
x=(c-(b*y))/a
```

البته در عبارت بالا و سورس و الگوریتم برنامه از اولین عبارت (ax+by=c) برای محاسبه‌ی مقدار x استفاده شده است. [3]

۱) a , b , c را دریافت کن

۲) b*b - 4*a*c ← delta

۳) اگر delta >= 0 آنگاه

i. (-b + sqrt) /(2 delta)) a ← x1

ii. (-b - sqrt) /(2 delta)) a ← x2

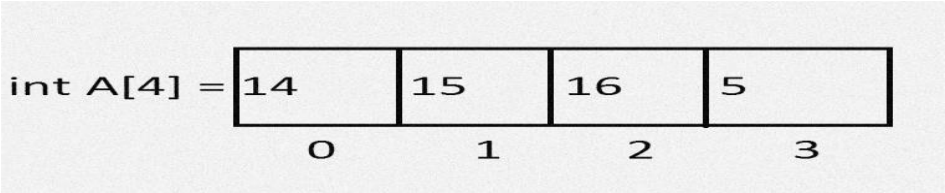
۴) x1 , x2 را نمایش بده

۵) در غیر این صورت

۶) پیغام " این معادله ریشه ندارد " را نمایش بده

۲-۳۳- الگوریتم جست و جوی خطی[8] در آرایه‌ها

توضیح: این الگوریتم برای پیدا کردن مقادیر مورد نظر ما بکار می‌رود. در واقع ساده ترین جستجوگر ممکن در زبان های برنامه‌نویسی می‌باشد. روش این الگوریتم روش مقایسه است بطور مثال در یک آرایه ده عضوی تا زمان پیدا کردن عدد مورد نظر، آن عدد را با تمام خانه های آرایه مقایسه می‌کند و این موضوع یکی از معایب آن به حساب می‌آید زیرا اگر آرایه ما متشکل از چندین هزار خانه باشد تا یافتن آن مقدار باید عدد وارده را با تمام وخانه مقایسه کند بطوری‌که اگر آن را نیابد تمام چنیدن هزار خانه را یک دور با آن مقدار مقایسه می‌کند. [2]



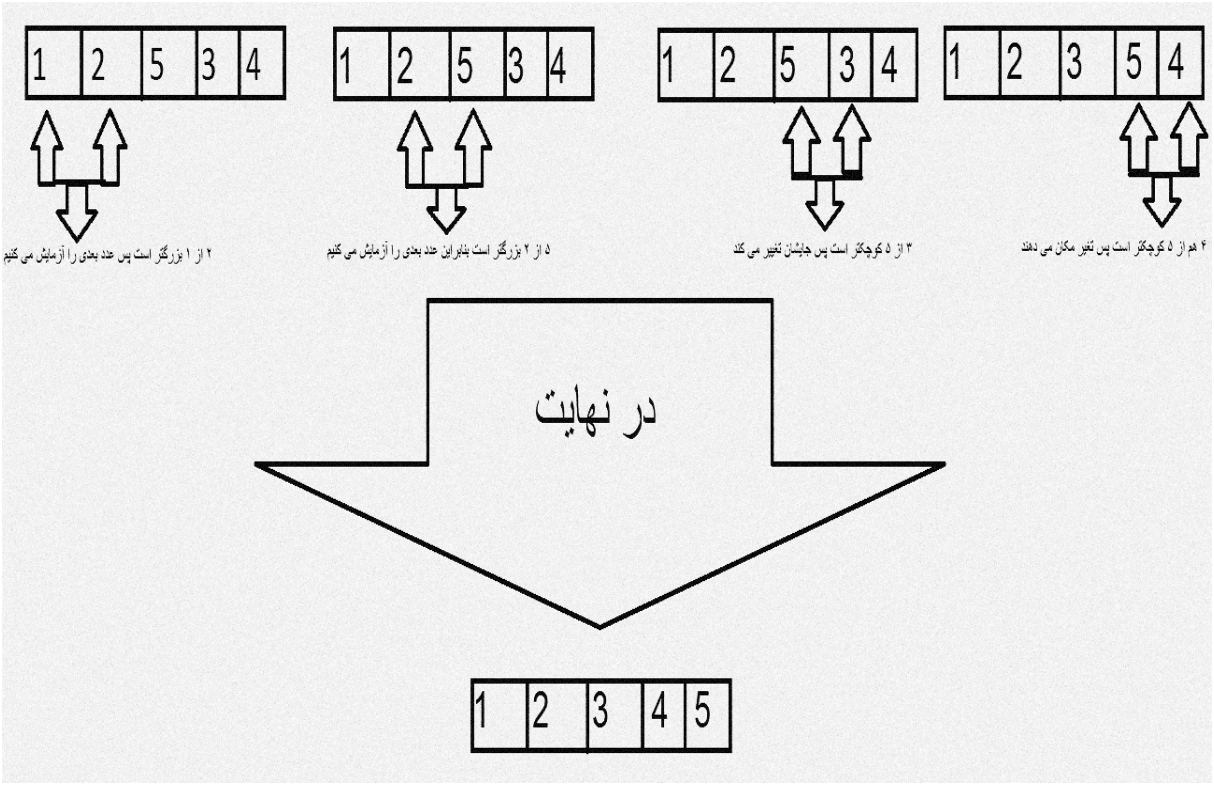
برای مثال اگر عدد مورد نظر ما ۵ باشد ابتدا چک می‌کند که آیا ۵ و ۱۴ برابر هستند یا نه در صورت نابرابری به خانه بعد می‌رود و آنقدر آین کار را انجام می‌دهد تا عدد را پیدا کند. [2]

۲-۳۴- الگوریتم مرتب‌سازی حبابی آرایه[9]

توضیح: این الگوریتم در واقع مجموعه‌ای از مقادیر را در قالب یک آرایه دریافت می‌کند و آن‌ها را بر اساس نیاز ما از کوچک به بزرگ یا از بزرگ به کوچک دسته‌بندی می‌کند. تعداد مراحلی که طول می‌کشد تا این الگوریتم داده‌هارا مرتب کند در صورتی که n همان تعداد اعداد باشد n-1 می‌باشد [2]

به تصویر زیر توجه کنید:

[8] Linear Search
[9] Bubble Sort



لیست زیر را در نظر بگیرید که می‌خواهیم به صورت صعودی (از کوچک به بزرگ) مرتب کنیم:

4 3 8 1 6 2

عنصر اول را با دومی‌مقایسه کرده و در صورتی که اولی بزرگتر باشد، جای آنها را تعویض می‌کنیم:

1 - 1) 4 3 8 1 6 2 → 3 4 8 1 6 2

همین کار را با عناصر دوم و سوم انجام می‌دهیم

1 - 2) 3 4 8 1 6 2 → 3 4 8 1 6 2

و همینطور عناصر سوم و چهارم:

1 - 3) 3 4 8 1 6 2 → 3 4 1 8 6 2

و الی آخر:

1 - 4) 3 4 1 8 6 2 → 3 4 1 6 8 2

1 - 5) 3 4 1 6 8 2 → 3 4 1 6 2 8

زمانی که به انتهای لیست رسیدیم، بزرگترین عنصر بین داده‌ها به انتهای لیست منتقل شده است.

در مرحله‌ی بعد، مجددا از ابتدا شروع کرده و تا انتها ادامه می‌دهیم. اما با توجه به این که به طور قطع عنصر n ام در جای اصلی خود قرار دارد، تا عنصر (n-1) ام پیش می‌رویم. در پایان این مرحله، بزرگترین عدد در لیست نامرتب باقیمانده، به انتهای آن منتقل می‌شود که دومین عدد از نظر بزرگی در کل لیست است: [2]

2 - 1) 3 4 1 6 2 8 → 3 4 1 6 2 8

2 - 2) 3 4 1 6 2 8 → 3 1 4 6 2 8

2 - 3) 3 1 4 6 2 8 → 3 1 4 6 2 8

2 - 4) 3 1 4 6 2 8 → 3 1 4 2 6 8

در هر مرحله، طول بازه‌ای که مرتب‌سازی روی آن انجام می‌گیرد، یک واحد کم شده و در نتیجه تعداد گام‌ها نیز یک گام کمتر می‌شود: [2]

3 - 1) 3 1 4 2 6 8 → 1 3 4 2 6 8

3 - 2) 1 3 4 2 6 8 → 1 3 4 2 6 8

3 - 3) 1 3 4 2 6 8 → 1 3 2 4 6 8

4 - 1) 1 3 2 4 6 8 → 1 3 2 4 6 8

4 - 2) 1 3 2 4 6 8 → 1 2 3 4 6 8

1) 1 2 3 4 6 8 → 1 2 3 4 6 8

در پایان این مراحل، لیست مرتب شده است.

۳۵-۲- الگوریتم جستجو سه‌تایی

توضیح: فرض کنید ما به دنبال بیشینه‌ی f هستیم و می‌دانیم این مقدار بین A و B قرار دارد. برای این که الگوریتم قابل اعمال باشند، یک مقدار x باید وجود داشته باشد به طوری که:

- به ازای همه‌ی مقادیر a و b، که $A \le a < b \le x$ داریم $f(a) < f(b)$
- به ازای همه‌ی مقادیر a و b، که $x \le a < b \le B$ داریم $f(a) > f(b)$

تابع اکید f(x) را روی بازه‌ی [l,r] در نظر بگیرید. دو نقطه ی m1 و m2 را در این بازه انتخاب می‌کنیم[2] .

بنابراین $l \le m1 < m2 \le r$ در این صورت سه حالت وجود دارد:

- اگر $f(m_1) < f(m_2)$، مقدار بیشینه نمی‌تواند درباره‌ی سمت چپ واقع شود-- $[l, m_1]$. این بدان معنی است که مقدار بیشینه در فاصله $[m_1, r]$ قرار دارد.
- اگر $f(m_1) > f(m_2)$، مقدار بیشینه نمی‌تواند درباره‌ی سمت راست واقع شود-- $[m_2, r]$. این بدان معنی است که مقدار بیشینه در فاصله $[l, m_2]$ قرار دارد.
- اگر $f(m_1) = f(m_2)$، این جستجو باید در بازه‌ی انجام شود. این حالت را می‌تونیم به هر یک از حالت‌های قبل برای ساده شدن رویه نسبت دهیم.

این فرایند تا زمانی ادامه پیدا می‌کند که بازه‌ی حاصل از مقدار ثابت از پیش تعیین شده کوچکتر شود.

انتخاب نقاط m1 و m2

$$m1 = l + (r-l)/3$$
$$m2 = r - (r-l)/3$$

۳۶-۲- الگوریتم جستجوی عمق محدود

توضیح: جستجوی عمق محدود، یک جستجوی ناآگاهانه است. این جستجو دقیقاً مشابه جستجوی اول-عمق عمل می‌کند، اما با تحمیل کردن یک محدودیت حداکثری روی عمق جستجو، از مشکلاتی که مربوط به تمامیت اول-عمق است، جلوگیری می‌کند. حتی اگر جستجو هنوز هم بتواند یک رأس فراتر از آن عمق گسترش دهد، نمی‌تواند چنین کاری را انجام دهد و درنتیجه، مسیرهای با عمق نامحدود را دنبال نمی‌کند یا به عبارتی در حلقه‌ها گیر نمی‌افتد؛ بنابراین، جستجوی با عمق محدود در صورتی جواب را پیدا خواهد نمود که این جواب در فاصله اولین سطح تا عمق محدود قرار داشته باشد، که این محدودیت حداقل تمامیت را روی تمامی گراف‌ها تضمین می‌کند. [2]

۱) رأسی را که الگوریتم باید از آن شروع کند و همچنین حداکثر عمق جستجو را مشخص کنید.

۲) چک کنید که آیا رأس کنونی حالت هدف است :

i. اگر نیست: هیچ کاری انجام ندهید.

ii. اگر پاسخ مثبت است: از الگوریتم خارج شوید.

۳) چک کنید که آیا رأس کنونی در عمق کمتر از حداکثر عمق جستجو قرار دارد :

i. اگر چنین نیست: هیچ کاری انجام ندهید.

ii. اگر پاسخ مثبت است.

۴) رأس را بسط دهید و تمامی‌نودهای زیرمجموعه آن را در پشته ذخیره کنید.

۵) DLS را به طور بازگشتی برای تمامی‌رئوس موجود در پشته فراخوانی کنید و به مرحله ۲ برگردید. [2]

۳۷-۲- الگوریتم جستجوی پرشی

توضیح: یک الگوریتم برای پیدا کردن یک کلید در یک آرایه مرتب است. این الگوریتم با چک کردن همه آیتم هایLkm ، جایی که l آرایه است و k عضو اعداد طبیعی و m اندازه بلوکی هست که در آن کلید را جستجو می‌کند و این الگوریتم ادامه میدهد این جستجو را تا زمانی که یک آیتم پیدا شود که از کلید جستجو بزرگتر باشد. در آن صورت از یک مرحله قبلش به اندازه m یکی یکی چک می‌کند و در صورت وجود کلید آن را پیدا می‌کند. [2]

مقدار بهینه m رادیکال n است، جایی که n اندازه آرایه l است. در هر دو مرحله الگوریتم به اندازه $n\sqrt{}$ پیش می‌رود پس این الگوریتم در $O(\sqrt{n})$ اجرا میشود که این الگوریتم از الگوریتم خطی بهتر و از الگوریتم دو دویی بدتر است. اما مزیت این الگوریتم به الگوریتم دودویی این است که در این الگوریتم فقط یک بار به عقب بر میگردد ولی در الگوریتم دودویی می‌تواند تا log n بار به عقب برگردد و این در مواقعی که برگشت به عقب موثرتر است از به جلو پیش رفتن خیلی اهمیت دارد. [2]

- ورودی: یک لیست مرتب ,L که طول آن آرایه n و کلید مورد نظر s است.
- خروجی: محل قرار گرفتن s در ,L یا هیچی اگر s موجود نباشد درL .

۳۸-۲- الگوریتم مرتب سازی تعویضی

توضیح: در الگوریتم‌های مرتب سازی قصدمان این است که اعداد یک آرایه ی عددی را به صورت صعودی یا نزولی مرتب کنیم. [2]

الگوریتم های مرتب‌سازی آرایه انواع مختلفی دارند مانند: حبابی، تعویضی، shell ، quick sort و.... [2]

در این الگوریتم مرتب‌سازی هر اندیس آرایه با اندیس‌های بعد از خود مقایسه می‌شود و طبق شرط مقایسه‌ی مان اعداد آرایه را مرتب می‌کنیم. [2]

شرط مقایسه:

صعودی: اگر این عدد از این اندیس آرایه بزرگتر از اعداد بعد از خود بود آن ها را با هم تعویض کن. [2]

نزولی: اگر این عدد از این اندیس آرایه کوچکتر از اعداد بعد از خود بود آن ها را با هم تعویض کن.

فرض کنید آرایه‌ی ما این باشد:

```
int A[5]={3,0,1,2,5};
```

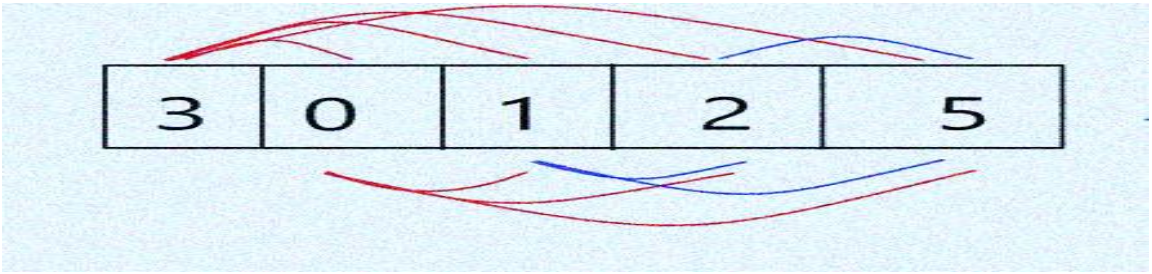
روش مرتب‌سازی به این صورت خواهد بود:

گام اول: اندیس صفرم یعنی ۳ ابتدا با ۰ بعد با ۱ سپس با ۲ و سپس با ۵ مقایسه می‌شود

گام دوم: اندیس یک یعنی ۰ با ۱ و ۲ و ۵ مقایسه می‌شود

و در گام های بعدی هم به همین صورت ادامه خواهد داشت.

در تصویر می‌توانید بهتر متوجه شوید:



مقایسه‌ها با خط نمایش داده شده است

۲-۳۹- الگوریتم یک رشته طولانی دریافت کند و تعداد کلمه and در این رشته را مشخص کند

توضیح: رشته را از ورودی دریافت می‌کنیم و سپس متغییری رابرابر با رشته کلمه and مشخص می‌کنیم همچنین یک متغییر برای شمارش طول رشته ای دریافتی انتخاب می‌کنیم.

تاجایی که I=0 بود و اگر از رشته ی وارد شده به مقدار I ، کلمه مورد جستجو (and) بود و به متغییر شمارنده یک عدد اضافه کن نمایش بده متغییر را در غیر اینصورت تکرار کنید.

۲-۴۰- الگوریتم پنجاه اسم را از ورودی دریافت و تعدادی افرادی که اسم آنها hadi است چاپ کند

توضیح: یک متغییر برای تکرار (حلقه) تعریف و یکی برای شمارش تعداد کلمه تعریف می‌کنیم S=0 .و یک متغییر نیز برای شرط می‌کنیم تکرار کن از I=1 تا جایی که کوچکتر از ۵۰ بود کلمه از ورودی X دریافت کن [2]

اگر X= hadi کلمه ورودی برابر با کلمه ی hadi بود به حلقه و شمارنده s=s+1 یک عدد اضافه کن و چاپ کن S (تعداد) در غیر اینصورت تا به پایان به حلقه ادامه بده. [2]

۲-۴۱- الگوریتم چاپ اعداد ۱تا ۵ به شکل حروف

توضیح: عددی را از ورودی دریافت کرده سپس شرط می‌گذاریم که اعداد اگر A==1 بود در اینصورت برو پایان چاپ کن one در غیر اینصورت اگر a==2 بود چاپ کن Two و در غیر اینصورت اگر a==3 برو پایان چاپ کن tree و اگر a==4برو پایان چاپ کن Four و اگر هم تمامی‌موارد نبود چاپ کن Fiveو از الگوریتم خارج شوید. [2]

۲-۴۲- الگوریتم برای بررسی آدرس ایمیل از نوار آدرس

توضیح: ایمیلی را از ورودی دریافت می‌کنیم و شرط گذاشته که تک تک رشته‌ها را جداکرده اگر در رشته دریافتی کارکتر (.) و همچنین اگر کاراکتر (@) بود در خروجی چاپ شود که ایمیل صحیح می‌باشد. در غیر اینصورت برو به پایان و چاپ شود که ایمیل صحیح نیست. [2]

۲-۴۳- الگوریتم نام کاربری غیر مجاز XX,admin

توضیح: کلمه‌ی را از ورودی دریافت می‌کنیم و تک تک کلمات را جداکرده و شرط می‌گذاریم که اگر کلمه‌ی ورودی برابر با XX و یا اگر کاراکتر ورودی برابر با کلمه Admin بود چاپ کنید کاراکتر غیر مجاز و در غیر اینصورت برو به پایان و در خروجی چاپ شود کاراکتر غیر مجاز بوده. [4]

۲-۴۴- الگوریتم تبدیل تاریخ میلادی به شمسی

توضیح: با استفاده از تابع jalali-To-grgrain استفاده می‌شود.

تاریخ را از Gdate خود سیستم دریافت می‌کنیم و سپس تاریخ دریافتی را به وسیله تابع بالا به شمسی تبدیل و در خروجی چاپ گردد. [4]

۲-۴۵- الگوریتم دنباله‌ی اعداد فیبوناچی

توضیح: بسیاری از فرآیندهای طبیعی از جمله ترکیب ساختار بدن موجودات زنده نظم مشخصی دارند و از دنباله‌ی اعدادی تبعیت می‌کنند که امروزه با نام دنباله‌ی اعداد فیبوناتچی[1] شناخته می‌شود. مشهورترین خاصیت این اعداد نسبت دو جمله‌ی متوالی آنها به ازای جملات بزرگ دنباله است که به عدد طلایی مشهور است .

تعریف: دنباله‌ی اعداد فیبوناچی روی اعداد حسابی به صورت زیر تعریف می‌شود :

Fn=⌈⌊|Fn−1+Fn−210n>1n=1n=0

همانگونه که از تعریف مشخص است، جملات این دنباله از جمع دو جمله‌ی قبلی آن با شروع از دو مقدار صفر و یک به دست می‌آید: [4]

[1] Fibonacci 0

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,...

۱) شروع

۲) 0 ← f 0

۳) 1 ← f 1

۴) f 0 و f 1 را نمایش بده

۵) 1 ← c

۶) مادامی که c<9 تکرار کن

{

i. sum ← f 0 + f 1

ii. Sumرا نمایش بده

iii. f1 ← f 0

iv. Sum ← f1

v. c+1 ← c

}

۷) پایان

۴۶-۲- الگوریتم ریش - انتگرال

توضیح: الگوریتم Risch برای تابع اولیه انتگرال استفاده شده است. این‌ها توابعی هستند که شامل ترکیب تابع‌های نمایی، لگاریتم‌ها، رادیکال‌ها، مثلثات و چهار عمل اصلی علم حساب (ضرب و جمع و تفریق و تقسیم) می‌باشند .

انتگرال نامعینِ یک تابع منطقی، تابعیست منطقی و یک عدد محدود از ضربِ تعداد محدودی از ضرب‌های ثابت لگاریتم تابع منطقی است[4] .

Liouville مسئله حل شده توسط الگوریتمRisch را به صورت فرمول درآورد Liouville . به روش تحلیلی اثبات کرد که اگر یک راه حل ابتدایی یا اولیه که اسمش g باشد وجود داشته باشد، برای معادله f = g ' آنگاه برای ثابت αi و تابع اصلی ui و v پاسخ به شکل زیرمی‌باشد.

$$g = v + \sum_{i<n} \alpha_i \ \ln(u_i)$$

Risch روشی ساخت که اجازه میداد شخص فقط نگرانِ مجموعه‌ای محدود از توابع اولیه به صورتِ فرمولِ لیوویل باشد.

فراستِ الگوریتم رِیش، از رفتارِ توابع نمایی و لگاریتمی‌در مشتق گیری می‌آید. برای تابع f eg که f و gتوابع متغیری هستند که ما داریم. [4]

$$(f \cdot e^g)' = (f' + f \cdot g') \cdot e^g,$$

لذا اگر eg در نتیجه انتگرال نامعین بود، باید انتظار داشت که داخلِ انتگرال نیز باشد همانگونه که:

$$(f \cdot \ln^n g)' = f' \ln^n g + nf\frac{g'}{g} \ln^{n-1} g$$

سپس اگر lnng در نتیجه انتگرال باشد آنوقت فقط توان‌های کمی‌از لگاریتم می‌توان انتظار داشت.

۴۷-۲- الگوریتم زیر مجموعه

توضیح: برای حل این مسئله می‌تونیم از یک الگوی باینری که تعداد صفر و یک‌های اون برابر n هست استفاده کنیم. مثلا برای یک مجموعه‌ی ۳ عضوی می‌تونیم از الگوی زیر استفاده کنیم: [4]

000

001

010

011

100

101

110

111

به طوری که به ازای هر عدد ۱، عضو مورد نظر نمایش داده شود. بدین ترتیب همه‌ی زیر مجموعه‌های مجموعه نمایش داده می‌شود. [4]

برای انجام این کار باید اعداد ۱ تا ۲ $2^{(n+1)}-1$ رو به صورت عدد باینری داخل یک آرایه قرار می‌دیم. بعد داخل حلقه بررسی می‌کنیم و در صورتی که تعداد ۰ و ۱های اعضای آرایه برابر تعداد اعضای مجموعه باشد به ازای هر عدد ۱، عضو مربوطه نمایش داده شود. [4]

۴۸-۲- الگوریتم کد ملی

توضیح: کد ملی شماره‌ای است ۱۰ رقمی که از سمت چپ سه رقم کد شهرستان محل صدور شناسنامه، شش رقم بعدی کد منحصر به فرد برای فرد دارنده شناسنامه در شهرستان محل صدرو و رقم آخر آن هم یک رقم کنترل است که از روی ۹ رقم سمت چپ بدست می‌آید. برای بررسی کنترل کد کافی است مجدد از روی ۹ رقم سمت چپ رقم کنترل را محاسبه کنیم. [4]

از آنجایی که درسیستم کد ملی معمولا قبل از کد تعدادی صفر وجود دارد.(رقم اول و رقم دوم از سمت چپ کد ملی ممکن است صفر باشد) و در بسیاری از موارد ممکن است کاربر این صفرها را وارد نکرده باشد و یا نرم‌افزار این صفرها را ذخیره نکرده باشد بهتر است قبل از هر کاری در صورتی که طول کد بزرگتر مساوی ۸ و کمتر از ۱۰ باشد به تعداد لازم (یک تا دو تا صفر) به سمت چپ عدد اضافه کنید. ساختار کد ملی در زیر نشان داده شده است. [4]

ساختار کد ملی										
ارقام کد	رقم کنترل	۹ رقم سمت چپ کد ملی								
موقعیت	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰

۱) برای محاسبه رقم کنترل از روی سایر ارقام، هر رقم را در موقعیت آن ضرب کرده و حاصل را با هم جمع می‌کنیم.

۲) مجموع بدست آمده از مرحله یک را بر ۱۱ تقسیم می‌کنیم

۳) اگر باقیمانده کمتر از ۲ باشد، رقم کنترل باید برابر باقیمانده باشد در غیر اینصورت رقم کنترل باید برابر یازده منهای باقیمانده باشد

مثال: آیا کد ۷۷۳۱۶۸۹۹۵۱ یک کد ملی معتبر است؟برای این منظور کد

ساختار کد ملی										
ساختار کد	رقم کنترل	۹ رقم سمت چپ کد ملی								
ارقام کد	۱	۵	۹	۹	۸	۶	۱	۳	۷	۷
موقعیت	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
محاسبه حاصل ضرب		۱۰	۲۷	۳۶	۴۰	۳۶	۷	۲۴	۶۳	۷۰

حاصل جمع ضرب ارقام ۲ الی ۱۰ را در موقعیت آنها محاسبه می‌کنیم

7*10+7*9+3*8+1*7+6*6+8*5+9*4+9*3+5*2=313

313÷11=28 و R=5

چون باقیمانده برابر ۵ و بزرگتر مساوی ۲ است پس باید رقم کنترل این کد برابر ۶ (یازده منهای ۵ برابر ۶)باشد .

با دقت در کد متوجه می‌شویم که رقم کنترل ورودی برابر ۱ است پس کد مورد نظر به عنوان یک کد معتبر قابل قبول نیست.

۴۹-۲- الگوریتم شابک کتاب

توضیح: هر ویراست از یک کتاب دارای شماره شابک خاص خود است و این شماره دارای ۱۰ تا ۱۳ رقم است و پنج بخش دارد:

اگر شماره ۱۳رقمی‌باشد دارای یک عددِ ۹۷۸ یا ۹۷۹ است.هر انتشارات دارای شماره خاص خود است که در شابک گنجانده می‌شود .

هر کتاب دارای شماره خاص خود است. [4]

نویسه وارسی (عدد کنترل) با استفاده از روش زیر محاسبه می‌شود:

۱) هریک از نویسه‌های شناسه شابک (از چپ به راست) در وزنی که از ۱۰ تا ۲ به‌ترتیب برای هریک از آنها در نظر گرفته می‌شود، ضرب می‌شود. [4]

۲) حاصل نتایج ضرب‌ها با هم جمع می‌شود.

۳) نویسه وارسی با استفاده از حاصل نتایج ضرب در مبنای ۱۱ تعیین می‌شود (یعنی برای مقدار مجموع حاصل‌ضرب‌های برابر با ۱۰، نویسه X استفاده می‌شود.([4]

مثلاً برای:

ISBN 600-5825-00-8

۱۰×۶ + ۹×۰ + ۸×۰ + ۷×۵ + ۶×۸ + ۵×۲ + ۴×۵ + ۳×۰ + ۲×۰ = ۶۰ + ۰ + ۰ + ۳۵ + ۴۸ + ۱۰ + ۲۰ +

۰ + = ۰ 173

حاصل مضرب کامل قبلی «۱۱ × ۱۵ = ۱۶۵» ۱۱ است.

باقی‌مانده تقسیم ۱۷۳ بر ۱۱ عدد ۸ می‌شود. به عبارت دیگر: ۱۷۳ - ۱۶۵ = ۸ عدد کنترل شابک این مثال ۸ است که سمت راست عدد آمده‌است. [4]

برای اعتبارسنجی شابک ۱۰ رقمی‌باید تمام اعداد را از سمت چپ به ترتیب در اعداد ۱۰ تا ۱ ضرب کرده و سپس حاصل هرکدام را با هم جمع کنیم. به عنوان مثال شابک ۰-۳۰۶-۴۰۶۱۵-۲ را به صورت زیر اعتبارسنجی می کنیم [4] .

$$\begin{aligned} s &= (0\times 10)+(3\times 9)+(0\times 8)+(6\times 7)+(4\times 6)+(0\times 5)+(6\times 4)+(1\times 3)+(5\times 2)+(2\times 1)\\ &= 0+27+0+42+24+0+24+3+10+2\\ &= 132 = 12\times 11 \end{aligned}$$

همانطور که می‌بینید حاصل محاسبه فوق عدد ۱۳۲ شده است و اکنون کافیست این عدد را بر عدد ۱۱ تقسیم کنیم. چنانچه باقیمانده صفر شود بعنی شابک مورد نظر صحیح است و در غیر اینصورت اشتباه می‌باشد.

۵۰-۲- الگوریتم تست شماره حساب بانک ملت

توضیح: تمامی کارت‌های اعتباری بانکی شامل الگوریتمی بین المللی هستند فرمول به این شکل است که ابتدا از رقم اول ۲ تا ۲تا جدا می کنید و آنها را در ۲ ضرب می کنید، در صورتی که این ارقام از ۱۰ بیشتر باشند آنها را از ۹ کم می کنید، نتایج را ذخیره کنید، حال مابقی ارقام که از ۱۰ بیشتر نبودند را نیز ذخیره کنید، و ارقامی که در این عملیات شرکت نداشتند را ذخیره کنید و ارقام ذخیره شده را جمع کنید، حاصل باید مضربی از ۱۰ باشد. [4]

مثال:

6104 3370 2926 0526

این یک کارت وَلید بانک ملت است

```
6 1 0 4 3 3 7 0 2 9 2 6 0 5 2 6
x2 x2 x2 x2 x2 x2 x2 x2
12-9 0 6 14-9 4 4 0 4
3 1 0 4 6 3 5 0 4 9 4 6 0 5 4 6
3+1+0+4+6+3+5+0+4+9+4+6+0+5+4+6=60
60 Mod 10 = 0
```

کارت معتبر است

۵۱-۲- الگوریتم شماره حساب

توضیح: این فرمول یک عدد را در برابر رقم تطبیق آن درستی یابی می‌کند، که عموماً به بک شماره حساب پاره ای به منظور تولید شماره حساب کامل اضافه می‌شود. این شماره حساب باید تست زیر را پاس کند:

۱) با شروع از اولین رقم سمت راست و حرکت به سمت چپ، یکی در میان رقم‌های شماره زوج را دو برابر کند.

۲) ارقام اعداد دو برابر شده را با اعدادی که دو برابر نشده اند جمع کند.

۳) اگر جواب جمع در پیمانه‌ی ۱۰ صفر شود، این شماره حساب درست می‌باشد، در غیر این صورت اعتبار ندارد.

Account number	۷	۹	۹	۲	۷	۳	۹	۸	۷	۱	x
Double every other	۷	۱۸	۹	۴	۷	۶	۹	۱۶	۷	۲	X
Sum of digits	۷	۹	۹	۴	۷	۶	۹	۷	۷	۲	=۶۷

رقم تطبیق (x) بوسیله ی محاسبه‌ی ضرب جمع ارقام در ۹ در پیمانه ۱۰ بدست می‌آید .

به بیان:Layman

۱) مجموع ارقام ۶۷ را محاسبه کن(۶۷).

۲) در ۹ ضرب کن (۶۰۳).

۳) رقم آخر را بگیر(۳).

۴) نتیجه رقم تطبیق می‌باشد.

روش جایگزین: رقم تطبیق بوسیله‌ی محاسبه تفریق رقم اول مجموع ارقام از ۱۰ بدست می‌آید.

۱) مجموع ارقام ۶۷ را محاسبه کن(۶۷).

۲) رقم یکان آن را در نظر بگیر(۷).

۳) رقم یکان را از ۱۰ کم کن.

۴) نتیجه ۳ می‌شود که رقم تطبیق است.

همه‌ی این اعداد ۷۹۹۲۷۳۹۸۷۱۰ , ۷۹۹۲۷۳۹۸۷۱۱ , ۷۹۹۲۷۳۹۸۷۱۲ , ۷۹۹۲۷۳۹۸۷۱۳ , ۷۹۹۲۷۳۹۸۷۱۴ ,۷۹۹۲۷۳۹۸۷۱۵ , ۷۹۹۲۷۳۹۸۷۱۶ , ۷۹۹۲۷۳۹۸۷۱۷ ,۷۹۹۲۷۳۹۸۷۱۸ , ۷۹۹۲۷۳۹۸۷۱۹ به صورت زیر درستی‌یابی می‌شوند.

۱) همه ارقام را یکی در میان از سمت راست دو برابر کن:

(۲×۱) = ۲ , (۲×۸) = ۱۶ , (۲×۳) = ٦ , (۲×۲) = ٤ , (۲×۹)

۲) مجموع همی‌ارقام را محاسبه کن (ارقام داخل پرانتز جواب ضرب های مرحله ۱ هستند(x + :)رقم تطبیق)

۷+(۸+۱)+۹+(٤)+۷+(٦)+۹+(٦+۱)+۷+(۲) = x + 67.

۳) اگر مجموع مضربی از ۱۰ بود، شماره حساب احتمالاً معتبر می‌باشد. ۳ تنها رقم معتبر است که مجموع ۷۰ را تولید می‌کند که مضربی از ۱۰ می‌باشد.

۴) بنابراین همه شماره حساب‌ها نا معتبر هستند بجز ۷۹۹۲۷۳۹۸۷۱۳ که دارای رقم تطبیق درست می‌باشد.

۵۲-۲- الگوریتم تبدیل حسابهای بانک‌ها به «شبا» و بالعکس

توضیح: برای ساخت ۱۹ رقم سمت راست «شبا» به ترتیبی که در الگوریتم محاسبه سیستم حساب هر بانک ذکر شده است عمل می‌شود. پس از ساخت این ۱۹ رقم، قواعد محاسبه «شبا» همانند دستورالعمل صورت می‌پذیرد. در این الگوریتم‌ها «شماره حساب بانک» به معنی ۱۹ رقم سمت راست «شبا» است. [4]

اولین رقم «شماره حساب بانک» به صورت زیر معین می‌شود:

۱) اگر شماره مربوط به حساب سپردهٔ متمرکز باشد معادل عدد صفر.

۲) اگر شماره مربوط به حساب سپردهٔ غیرمتمرکز (وابسته به شعبه) باشد معادل عدد ۱.

۳) اگر شماره مربوط به حساب تسهیلات متمرکز باشد معادل عدد ۲.

۴) اگر شماره مربوط به حساب تسهیلات غیرمتمرکز (وابسته به شعبه) باشد معادل عدد ۳.

در صورتی که نوع حساب مشخص نباشد، به طور پیش‌فرض حساب سپرده در نظر گرفته می‌شود.

به هنگام تبدیل «شماره حساب بانک» مندرج در «شبا» به شماره حساب داخلی بانک، طول آن باید ۱۹ رقم باشد. ارقام سمت چپ این شماره باید یکی از ارقام صفر تا چهار باشد. در غیر این صورت شمارهٔ حساب معتبر نیست. [4]

۵۳-۲- شماره استاندارد بین‌المللی پیایندها

توضیح: فرمت شماره سریال استاندارد بین‌المللی یک شماره هشت رقمی (حرفی) است که توسط اتصال‌ها و فواصل به دو قسمت چهار رقمی‌تقسیم می‌گردد. عدد آخر که می‌تواند از ۰ تا ۹ و یا X باشد «شماره کنترل» نامیده می‌شود. گزارش خوانش (توسط دستگاه‌های بارکد خوان) این شماره برای مثال ممکن است چیزی مانند ۵۹۵۵-۰۳۷۸ باشد که از طریق الگوریتم زیر به تطبیق و کنترل آن مبادرت می‌شود:

- محاسبه حاصل جمع هفت رقم ابتدایی شماره ISSN به ترتیب در حاصل ضرب در شماره‌های ۸، ۷، ۶، ۵، ۴، ۳ و ۲:

```
0*8 + 3*7 + 7*6 + 8*5 + 5*4 + 9*3 + 5*2= 0 + 21 + 42
+ 40 + 20 + 27 + 10= 160
```

- سپس ضریب عدد ۱۶۰ را به پیمانه ۱۱ محاسبه می‌کنیم:

پیمانه (۱۶۰ بر ۱۱) = ۱۶

- شما می‌بایست متناوباً این تقسیم را بر پایه ۱۱ انجام بدهید:

160 / 11 = 14.6

- سپس قدرمطلق باقی‌مانده (که در بالا شش است) ارزش شماره کنترل ما را وقتی که از میزان ۱۱ تفریق می‌شود، را مشخص می‌کند:

11 - 6 = 5

شماره کنترل ما پنج می‌شود. همچنین X نشان دهنده شماره ده در شماره کنترل می‌باشد.

همچنین طبق الگوی بالا یک سیستم چک کننده معتبر بودن ای اس اس وجود دارد است که از طریق آن می‌توان معتبر بودن یا جعلی بودن ای اس اس ان را تشخیص داد. [4]

۵۴-۲- الگوریتم بارکد

توضیح: این نوع بارکد با عنوان کد جهانی محصول یا UPC شناخته می شود. در ابتدا دانش آموزان با الگوریتم آشنا می شوند و سپس به تعیین اعتبار عدد می‌پردازند. سپس دانش آموزان باید برای تعیین یک رقم چک کننده برای یک بارکد تلاش کنند. در سیستمUPC، باید باقی مانده‌ی شماره‌ی UPC بر ۱۰ صفر باشد. [4]

این سیستم، از فاکتور ۳، برای رقم‌هایی که در جایگاه زوج قرار دارند، استفاده می‌کند، به این معنی که رقم‌های جایگاه زوج در ۳ ضرب می‌شوند. به عنوان اولین مثال از UPC داده شده در زیر که مربوط به فیلم شگفت انگیزان می‌باشد، استفاده کنید: [4]

0-24425-86936-7

برای بررسی این شماره، مراحل زیر را دنبال کنید:

۱) هر رقمی که در جایگاه زوج قرار دارد، در عدد ۳ ضرب می‌شود (شمارش را از سمت راست به چپ انجام می‌دهیم). ارقامی که در جایگاه فرد قرار دارند، در ۱ ضرب می‌شوند.

3(7)+1(8)+3(6)+1(9)+3(3)+1(6)+3(2)+1(4)+3(4)+1(4)+3(4)+1(2)+3(5)+1(0)

۲) نتایج را با یکدیگر جمع کنید.

0+15+2+12+4+6+6+9+9+18+8+21=110

۳) اعتبار این شماره را با تقسیم آن بر عدد ۱۰ تعیین نمایید.

110تقسیم بر ۱۰ دارای باقی مانده‌ی صفر می‌باشد (۱۱=۱۱۰/۱۰). پس این شماره‌یUPC، معتبر است.

ممکن است از ضرایب فاکتور گرفته شود، به این صورت که ابتدا رقم‌های موجود در جایگاه زوج را با یکدیگر جمع و سپس در ۳ ضرب کنند، رقم‌های موجود در جایگاه فرد را نیز جمع کرده و در ۱ ضرب کنند. [1]

سپس شماره‌ی UPC مقابل را مثال بزنید:

7-96714-78601-y در این جا y ، رقم چک کننده است. با استفاده از فرآیند فوق، دانش آموزان باید رقم چک کننده را تعیین کنند. مجموع نتایج ۱۱۲ است. پس رقم چک کننده باید ۸ باشد، زیرا باقیمانده‌ی تقسیم (۸+۱۱۲) بر ۱۰، ۰ است. در تمرین شماره ی ۱، باید رقم چک کننده را برای ۲ شماره‌ی UPC به دست آورند. [1]

نوع دیگری از سیستم‌های بارکد، شماره‌ی استاندارد بین المللی کتاب یا ISBN است.ISBN، یک عدد ده رقمی است که شامل بلوک‌های عددی با معانی متفاوت است. این شماره شامل چهار قسمت می‌باشد که با خط تیره یا فاصله از هم جدا شده‌اند.

اولین قسمت شماره که نشان‌دهنده‌ی زبان یا کشور است، شناسه‌ی گروه نام دارد و حداکثر ۵ رقمی است. [1]

بخش دوم شماره، نشان‌دهنده‌ی ناشر و حداکثر ۷ رقمی است.

سومین بخش شماره، نشان‌دهنده‌ی تجدید چاپ یا تعداد چاپ است و بیش از ۶ رقم نیست .

قسمت نهایی، رقم چک کننده است.

به یاد آورید که یک شماره‌ی ۱۰ رقمی داریم که رقم آخر آن، رقم چک کننده است. بنابراین سه قسمت اول شماره می‌باید در مجموع، یک عدد نه رقمی را تشکیل دهند. رقم‌های صفر، به عنوان پر کننده‌های فضا در ابتدای عدد، در صورتی که تعداد رقم‌ها به تعداد کافی نباشد، استفاده می‌شوند. شکل زیر، نمونه‌ای از شماره‌ی ISBN را نشان می‌دهد. [2]

در این سیستم، رقم چک کننده، به روشی متفاوت با روش UPC محاسبه می‌شود. به این صورت که اولین رقم در ۱۰ ضرب می‌شود، رقم دوم در ۹، رقم سوم در ۸ و این فرایند را ادامه می‌دهیم تا رقم نهم که در ۲ ضرب می‌شود. سپس مجموع نتایج را محاسبه می‌کنیم. این سیستم پیمانه‌ی۱۱ نامیده می‌شود، به این معنی که مجموع نتایج نه رقم اول به علاوه‌ی رقم چک کننده باید مضرب ۱۱ باشد. [3]

یکی از مشکلات این روش این است که ممکن است رقم چک کننده ۱۰ باشد، در حالی که ما فقط می‌توانیم از ارقام ۰ تا ۹ برای این منظور استفاده کنیم و بنابر این X به جای رقم چک کننده نوشته می‌شود .(این X نشان‌دهنده ی شماره ۱۰ رومی است.) تمرین‌های شماره‌ی ۳ تا ۶، مشخصات مربوط به ISBN می‌باشند.

کارت‌های اعتباری، از یک سیستم بلوک‌های شماره‌ای، مشابه سیستم ISBN استفاده می‌کنند. یک تفاوت واضح این است که در این جا حداکثر طول شماره ۱۹ رقم است، اگرچه بسیاری از شماره‌ها بین ۱۳ تا ۱۶ رقم می‌باشند.

صادر کننده	شناسه	طول شماره ی کارت
Diner's Club/ Carte Blanche	۳۰۰xxx—305xxx ۳۶xxxx, 38xxxx	۱۴
پست آمریکا	۳۴xxxx, 37xxxx	۱۵
Visa	۴xxxxx	۱۳،۱۶
MasterCard	۵۱xxxx—55xxxx	۱۶
Discover	۶۰۱۱xx	۱۶

اولین رقم شماره‌ی کارت اعتباری، شناسه‌ی صنعت اصلی یا MII[1] نام دارد و نشان‌دهنده‌ی گروه صادر کننده‌ی کارت است، همان طور که در جدول زیر نشان داده شده است. [4]

مقدار رقم MII	گروه صادر کننده
۰	ISO/TC 68 و صنایع وابسته
۱	خطوط هوایی
۲	خطوط هوایی و صنایع وابسته
۳	مسافرت وسرگرمی
۴	بانکداری و امور مالی
۵	بانکداری و امور مالی
۶	فروش و بانکداری
۷	سوخت
۸	ارتباطات و صنایع وابسته
۹	مراکز ملی

به عنوان مثال، هنگامی که شماره با ۳ شروع می‌شود، نشان دهنده‌ی گروه مسافرت و سرگرمی است. کارت‌های پست آمریکا در این گروه قرار می‌گیرند. کارت های صادر شده توسط شرکت‌های گاز با رقم ۷ شروع می‌شوند. کارت‌های Visa و MasterCard در گروه بانکداری و امور مالی قرار می‌گیرند (۴ و ۵). قسمت دوم شماره، شناسه‌ی صدور است که با احتساب رقم MII، شش رقمی است. [4]

[1] Major Industry Identifier [1]

شماره حساب از رقم هفتم شروع می‌شود و تا رقم یکی مانده به آخر ادامه می‌یابد. رقم آخر، رقم چک کننده است. [4]

روش استفاده شده جهت محاسبه‌ی رقم چک کننده، الگوریتم لوهن (پیمانه ی ۱۰) است که به نام دانشمند IBM، هانس پیتر لوهن، نام گذاری شده است. این الگوریتم به روش زیر عمل می‌کند:

ابتدا تمام رقم‌های جایگاه زوج را دو برابر کنید (ارقام از راست به چپ خوانده می‌شوند). مجموع نتایج گام اول و رقم‌های جایگاه فرد را که به شکل اولیه هستند، محاسبه کنید.

در صورتی که حاصل گام دوم، مضرب ۱۰ باشد، شماره حساب تأیید می‌شود.

قبل از این که به قسمت تمرین‌های مربوط به این بحث مراجعه کنید، جهت آشنایی بیشتر دانشجویان با الگوریتم لوهن، به تعیین صحت رقم چک کننده‌ی شماره حساب زیر بپردازید:

```
5314772685932112
```

مجموع به دست آمده توسط الگوریتم، ۱۰۱ می‌باشد که به روش زیر محاسبه شده است:

```
101 = 2(5) + 3 + 2(1) + 4 + 2(7) + 7 + 2(2) + 6 + 2(8)
+ 5 + 2(9) + 3 + 2(2) + 1 + 2(1) + 2
```

برای این که این شماره حساب تأیید شود، مجموع باید بر ۱۰ قابل قسمت باشد. اگر رقم چک کننده، ۱ بود، باقیمانده‌ی تقسیم حاصل جمع بر ۱۰، صفر می‌شد ولی چون رقم چک کننده ۲ است، مجموع بر ۱۰ قابل قسمت نیست. بنابراین، این شماره حساب معتبر نیست.

الگوریتم لوهن می‌تواند خطاهای ورود داده و اکثر جا به جایی‌ها را تشخیص دهد. دانشجویان باید این مبحث را ادامه دهند و تعیین کنند که این فرایند چگونه صورت می‌گیرد.

۵۵-۲- الگوریتم امضای دیجیتال

توضیح: یک تابع پنهانی در هم تصویب شده H را انتخاب کنید. بر روی طول کلیدL,N تصمیم بگیرید. این اندازه گیری اولیه قدرت پنهانی کلید است. دی اس اس اصلی ما را وادار می‌کند تا مضربی از ۶۴ بین ۵۱۲و ۱۰۲۴ باشد.

۱) یک بیت اولیه n را به گونه‌ای برگزینید که q . N کمتر یا مساوی با طول خروجی درهم باشد.

۲) یک بیت اولیه l با مدول p را به گونه‌ای انتخاب کنید که p-۱ مضربی از q باشد.

۳) عددی را به عنوانhp- = (g ۱/ q) برگزینید.

پارامترهای الگوریتم (p, q, g) ممکن است بین کاربران سیستم به اشتراک گذاشته شود. به ازای هر کاربر یک مجموعه از پارامترها به کلیدها تخصیص میابد. به ازای هر کاربر یک مجموعه از پارامترها به کلید تخصص میابد مرحله دوم کلیدهای عمومیو اختصاصی را برای یک کاربر مجزا محاسبه می‌کند.

۱) انتخاب x با روش‌های تصادفی

۲) محاسبه باقی‌ماندهgx = y

۳) کلید عمومی(p, q, g, y) و کلید خصوصی x است

۴) تولید یک کلید تصادفی k باید بعد از یکبار استفاده از بین رفته و دیگر مورد استفاده قرار نگیرد.

۵) سپس زوج مرتب امضا(r,s) به صورت زیر محاسبه می‌شوند.

R] = (g)k mod p) mod q s = [k-1(H(M) + xr)(mod q

(r,s) به پیام M الحاق شده و فرستاده می‌شود.

گیرنده M, و (r,,s,) را دریافت می‌کنند. مقادیر زیر را محاسبه می‌کند:

- w = (s')-1 mod q
- u1= [H(M')w] mod q
- u2= [(r')w] mod q
- v = (gu1y)u2) [mod p] mod q

اگر v=r, امضا معتبر است.

روش کار بدین شرح است که:

۱) فرستنده با استفاده از الگوریتم‌های هش PGP و متن خام اطلاعات (مثلا متن ایمیل) یک رشته HASH را ایجاد می‌نماید. که با نام Message Digest شناخته می‌شود

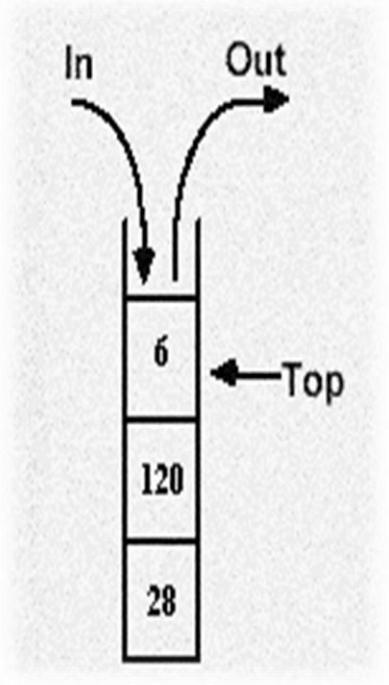
۲) رشته ایجاد شده توسط کلید خصوصی فرستنده رمزنگاری می‌شود. به عنوان امضای دیجیتال[۱۲]

۳) امضای دیجیتال به همراه اطلاعات ارسال می‌شود (می‌توان اطلاعات را نیز توسط PGP رمزنگاری کرد

۴) در سمت گیرنده این امضا با استفاده از کلید عمومی‌فرستنده رمزگشایی می‌شود. و Message Digest اولیه به دست می‌آید.

۵) اطلاعات اصلی نیز در صورت نیاز رمزگشایی شده و با استفاده از الگوریتم‌های هش PGP یک رشته HASH تولید می‌شود.

۶) در صورتی که HASH به دست آمده با Message Digest برابر بود به معنی این است که اطلاعات و فرستنده معتبر است.



۵۶-۲- الگوریتم تابع pop در پشته

توضیح: برای Pop کردن از پشته، داده‌ای که در اندیس Top قرار دارد برگردانده می‌شود و Top یک واحد کم می‌شود.

۱) عددی را دریافت می‌کنیم

۲) شرط گذاشته اگر پشته خالی بود Top =0 بود چاپ کند پشته خالی است به مرحله آخر برود

۳) در غیر اینصورت عدد را از پشته برمی داریم

۴) از بالاترین سطح پشته یک واحد کم کنیم

۵) عدد ۱ راچاپ کنید.

۵۷-۲- الگوریتم push در پشته

توضیح: اندیس عنصر بالای پشته Stack را نگه می‌دارد Top=n .دلالت بر خالی بودن پشته دارد .

[1] Digital Signature [2]

برای Push کردن یک داده جدید به پشته، Topیکی اضافه می‌شود و داده جدید در آرایه در اندیس Top درج می‌کند.

۵۸- الگوریتم اضافه کردن عدد به صف:

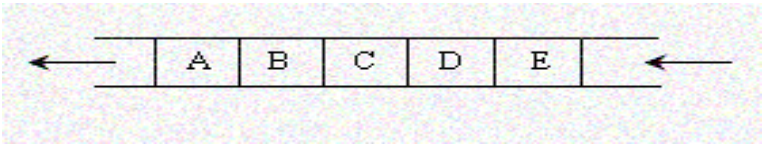
توضیح: برای اضافه کردن عدد به صف باید از انتهای صف یک واحد اضافه کنیم تا بتوانیم عدد را وارد کنیم.

صف دو اشاره گر Front که به ابتدای آرایه اشاره و Rear به انتهای آرایه اشاره می‌کند.

ازعمل اصلی Queue برای تست خالی بودن پراست در غیر اینصورت به Rear ما برابر با n بود چاپ کن پر است در غیر اینصورت به Raer یک واحد اضافه و عدد را وارد صف کنید.

صف لیست مرتبی است که عناصر در انتهای آن[۱۳] اضافه و از ابتدای آن[۱۴]حذف می‌شوند. به عبارت دیگر طول صف از انتهای آن افزایش و از ابتدای آن کاهش می‌یابد .

اولین عنصری که وارد صف می‌شود اولین عنصری است که از صف خارج می‌شود. بنابراین عناصر به همان ترتیبی که به صف اضافه می‌شوند از آن حذف می‌شوند. به همین دلیل به صف لیست FIFO[۱۵] نیز گفته می‌شود.



۵۸-۲- الگوریتم حذف کردن عدد از صف

توضیح: از عمل Quek برای پربودن یاخالی بودن استفاده می‌شود.

اگر Front برابربا Raer بود چاپ کنید خالی است در غیر اینصورت از ابتدای صف[۱۶] یک واحد اضافه کن و موقعیت Front یک خانه جلو آمده و عدد اولی حذف گردد. [1]

۵۹-۲- الگوریتم جستجوی پرتو محلی

توضیح: یکی از الگوریتم‌های فرا ابتکاری، الگوریتم جستجوی بیم یا پرتو محلی است که گراف را با استفاده از راس‌هایی که در یک مجموعه خاص احتمال وجود بیشتری دارند می‌پیماید. الگوریتم جستجوی اول سطح پیدا کردن کمترین فاصله بین راس شروع و پایان را در گراف بی‌وزن تضمین می‌کند ولی برای جستجو در فضاهای

[1] Rear [3]
[1] Front [4]
[1] first in, first out [5]

[1] Front [6]

بزرگ این کار تقریباً غیر عملی است چرا که حافظه‌ی بسیار زیادی مصرف می‌کند ممکن است قبل از این‌که به جواب برسد حافظه پر بشود.

الگوریتم بیم برای اینکه در حافظه صرفه‌جویی کند یک تابع h برای پیش‌بینی هزینه‌ی رسیدن به راس موردنظر از راس داده‌شده.همچنین از یک پارامتر به نامB (عرض بیم) استفاده می‌کند که نشان‌دهنده‌ی تعداد راس‌هایی است که در هر مرحله از الگوریتم جستجوی اول سطح ذخیره شده‌است. بنابراین زمانی که الگوریتم جستجوی اول سطح همه‌ی راس‌های مرزی (راس‌هایی که با نزدیک‌ترین راس ارتباط دارند) را ذخیره می‌کند الگوریتم بیم فقط راس‌های B با بهترین مقدار در هر مرحله از جستجو را ذخیره می‌کند. [1]

ایده اصلی این است که تابع h به الگوریتم اجازه می‌دهد که راس‌هایی انتخاب شوند که مسیر را به سمت راس مقصد راهنمایی کنند و پارامتر B باعث می‌شود که الگوریتم فقط این راس‌های مهم را ذخیره کند و از پر شدن حافظه قبل از رسیدن به هدف جلوگیری می‌کند.

برخلاف الگوریتم جستجوی اول سطح که از یک لیست[۱۷]استفاده می‌کند این الگوریتم راس‌هایی را که در حلقه‌ی بعد الگوریتم بسط داده می‌شود رادر[۱۸]ذخیره می‌کند؛ همچنین از یک جدول‌هش برای ذخیره‌ی راس‌هایی که دیده شده‌اند استفاده می‌کند شبیه[۱۹]درالگوریتم جستجوی اول سطح. الگوریتم بیم در ابتدا راس شروع را به BEAMوجدول‌هش اضافه می‌کند.

سپس در هر مرحله از حلقه‌ی اصلی از الگوریتم، راس‌های همسایه باراس‌های BEAM را به یک مجموعه که شامل راس‌های جانشین[۲۰]است اضافه می‌کند و سپس راس‌های B با بهترین مقدار از مجموعه‌ی جانشین را به BEAMو جدول‌هش اضافه می‌کند .

راس‌هایی که در حال حاضر در جدول‌هش قرار دارند به BEAM اضافه نمی‌شوند چرا که مسیر کوتاهتر برای آن راس پیدا شده.این فرایند ادامه دارد تا زمانی که راس مقصد پیدا شود، جدول‌هش پر شود، یا BEAM بعد از حلقه‌ی اصلی خالی شود. [1]

۶۰-۲- الگوریتم اسپیگوت

توضیح: یک نوع خاص از الگوریتم‌ها است که برای محاسبه ی مقدار یک ثابت ریاضی مانند π یا e استفاده می‌شود، که می‌تواند یک رشته ی خروجی از ارقام را بدون نیاز به استفاده ی مجدد از آنها تولید کند.

[17] open list
[18] BEAM
[19] close list
[20] successor

نام الگوریتم اسپیگوت توسط استنلی رابینوویتز و استان واگن ابداع شد که الگوریتم ابداعی آنها برای محاسبه عدد π برخی اوقات تحت عنوان اگوریتم اسپگوت برای "π " نام برده می‌شود.

الگوریتم اسپیگوت به این معنا که تعداد ارقام خواسته شده باید از قبل مشخص باشد. پیشرفت‌های بیشتر منجر به ابداع الگوریتم‌ای گردید که قابلیت محاسبه یک عدد دلخواه را بدون نیاز به محاسبه ارقام پیشین آن در وهله اول،دارا است:

فرمول بیلی-بوروین-پلوفه نمونه‌ای از این الگوریتم است.یک الگوریتم تجزیه ارقامی‌برای عدد π که ارقام را در مبنای ۶ تولید می‌کند.

این مثال نحوه ی کار کردن الگوریتم اسپیگوت را با محاسبه ی ارقام باینری لگاریتم طبیعی ۲ نشان می‌دهد (دنباله‌ی A068426 در OEIS) با استفاده از تساوی ذیل:

برای محاسبه ارقام باینری از جایگاه هشتم ما طرفین تساوی فوق را در ۲۷ ضرب می‌کنیم (با توجه به اینکه۷=-۸ ۱) سپس ما سری نا محدود را به دو قسمت تقسیم می‌کنیم: قسمت "head" که در آن توان ۲ بزرگتر یا مساوی ۰ است و قسمت "tail" که در آن توان ۲ منفی است

از آنجایی که ما فقط علاقه‌مند به بخش کسری این مقدار هستیم می‌توانیم هر مؤلفه سری در "head" را با مقدار زیر جایگزین کنیم.

با محاسبه هر کدام از این مقادیر و اضافه کردن آنها به سری کلی که در حال اجراست (که مجدداً در این سری کلی ما تنها علاقه‌مند به بخش کسری هستیم) به نتایج زیر می‌رسیم:

Sum of C mod 1	C = B / k	B = A mod k	A = 27-k	k
۰	۰	۰	۶۴	۱
۰	۰	۰	۳۲	۲
۳/۱	۳/۱	۱	۱۶	۳
۳/۱	۰	۰	۸	۴
۱۵/۲	۵/۴	۴	۴	۵
۱۵/۷	۳/۱	۲	۲	۶
۱۰۵/۶۴	۷/۱	۱	۱	۷

ما مقادیری را به بخش "tail" می‌افزاییم،با توجه به اینکه خطای تولید شده به موجب کوتاه کردن سری کمتر از مقدار نهایی است.

Maximum error	Sum of D	D = 1/k2k-7	k
۱۶/۱	۱۶/۱	۱۶/۱	۸
۳۶/۱	۱۴۴/۱۳	۳۶/۱	۹
۸۰/۱	۳۶۰/۳۷	۸۰/۱	۱۰

با اضافه کردن بخش "head" و تعداد اندکی از مقادیر نخستین بخش "tail" به نتایج زیر می‌رسیم:

بنابراین ارقام ۸ ام تا ۱۱ ام در بسط باینری(ln) 2ارقام ۱،۱،۰،۱ می‌باشند.توجه کنید که ما مقادیر ۷ رقم نخست باینری را محاسبه نکرده‌ایم – در واقع تمامی‌اطلاعات مربوط به آنها با استفاده از هم نهشتی(نظریه اعداد) در بخش سری "head" به صورت عمدی حذف شده است.

روش مشابهی می‌تواند برای محاسبه ی ارقام بسط باینری (ln) 2با شروع از جایگاه دلخواه n ام مورد استفاده قرار بگیرد. تعداد عبارت‌های موجود درقسمت "head" سری به صورت خطی با n افزایش میابد، اما چناچه یک روش کارآمدی با استفاده از به‌توان‌رساندن مدولار به کار گرفته شود پیچیدگی هرعبارت تنها با لگاریتم n افزایش میابد. دقت محاسبات،نتایج میانی و تعداد عبارت‌های گرفته شده از قسمت "tail" سری همگی مستقل از n می‌باشند و تنها وابسته به تعداد ارقام باینری هستند که در حال محاسبه است – یک واحد محاسباتی می‌تواند برای محاسبه حدودا ۱۲ رقم باینری صرف نظر از جایگاه شروع مورد استفاده قرار گیرد.

۶۱-۲- الگوریتم تقاطع خط و پاره خط‌ها

توضیح: برای محاسبه محل تقاطع دو خط کافیست معادله آن دو را حل کرده و در صورتی که جواب داشته باشند آن دو خط متقاطع اند. حال برای بررسی این که آیا دو پاره خط با هم متقاطع هستند یا نه باید به صورت زیر عمل کنیم :

۱) ابتدا محل تقاطع خطوط حامل پاره خط هارا در صورت وجود پیدا می‌کنیم.

۲) سپس بررسی می‌کنیم که آیا نقطه محاسبه شده بر روی یکی از پاره خط ها قرار دارد یا نه.

برای انجام قسمت ۱ کافیست معادله خط را با داشتن ۲ نقطه از آن بدست آورده و آن‌ها را حل کنیم. برای قسمت ۲ نیز می‌بایست پاره خط را با بردار نقطه تقاطع به مبدا انتقال دهیم و بررسی کنیم که آیا دو سر پاره خط در دو ربع متفاوت هستند یا خیر.

- حل مساله برای اعداد مختلط

توضیح: هر پاره خط در صفحه مختلط به وسیله ۲ عدد قابل نمایش است که همان ابتدا و انتهای پاره خط می‌باشد. مثلا ab پاره خطیست که عدد a یک سر آن و عدد b سر دیگر پاره خط است.حال برای پیدا کردن محل تقاطع ۲ پاره خط بایستی شرط‌هایی لازم و کافی برای یک نقطه دلخواه مانند X بیابیم که در صورت برقراری آن‌ها نقطه X روی خط حاصل از امتداد پاره خط ab قرار داشته باشد.حال فرض کنید نقطه X روی خط حاصل از امتداد ab باشد، در این صورت خواهیم داشت :

$$x-ab-a=(x-ab-a) \quad \bar{} \quad , \quad x-ab-a=(x-ab-a)$$

بنابراین :

$$(x-a)(b-a)=(b-a)(x-a),$$
$$(x-a)(\bar{b}-\bar{a})=(b-a)(\bar{x}-\bar{a})$$
$$(b-a)x-(b-a)x=(b-a)a-(b-a)a \qquad (1)$$
$$(1)\ (\bar{b}-\bar{a})x-(b-a)\bar{x}=(\bar{b}-\bar{a})a-(b-a)\bar{a}$$

حال فرض کنید نقطه x همزمان روی خط cd (خط حاصل از امتداد پاره خط cd) نیز قرار دارد.بنابراین مشاب‌ها خواهیم داشت :

$$x-cd-c=(x-cd-c) \quad \bar{} \quad , \quad x-cd-c=(x-cd-c)$$

بنابراین :

$$(x-c)(d-c)=(d-c)(x-c),$$
$$(x-c)(\bar{d}-\bar{c})=(d-c)(\bar{x}-\bar{c}),$$
$$d-a)x-(d-c)x=(d-c)c-(d-c)c \qquad (2)$$
$$(2)\ (\bar{d}-\bar{a})x-(d-c)\bar{x}=(\bar{d}-\bar{c})c-(d-c)\bar{c}$$

حال دو طرف معادله (۱) را در (d-c) ضرب می‌کنیم :

$$(d-c)(b^{-}-a^{-})x-(d-c)(b-a)x^{-}=((b^{-}-a^{-})a-(b-a)a^{-})(d-c) \quad (3)$$

$$(3)\ (d-c)(b^{-}-a^{-})x-(d-c)(b-a)x^{-}$$

$$=((b^{-}-a^{-})a-(b-a)a^{-})(d-c)$$

هم چنین دو طرف معادله (۲) را در (b-a) ضرب می‌کنیم :

$$(b-a)(d^{-}-c^{-})x-(d-c)(b-a)x^{-}=((d^{-}-c^{-})c-(d-c)c^{-})(b-a) \quad (4)$$

$$(4)\ (b-a)(d^{-}-c^{-})x-(d-c)(b-a)x^{-}=((d^{-}-c^{-})c-(d-c)c^{-})(b-a)$$

حال از تفاضل معادله (۳) و (۴) خواهیم داشت :

$$(d-c)(b^{-}-a^{-})x-(b-a)(d^{-}-c^{-})x=((b^{-}-a^{-})a-(b-a)a^{-})(d-c)-((d^{-}-c^{-})c-(d-c)c^{-})(b-a),\ (d-c)(b^{-}-a^{-})x-(b-a)(d^{-}-c^{-})$$

$$x=((b^{-}-a^{-})a-(b-a)a^{-})(d-c)-((d^{-}-c^{-})c-(d-c)c^{-})(b-a),$$

که مقدار x بدست می‌آید :

$$x=((b^{-}-a^{-})a-(b-a)a^{-})(d-c)-((d^{-}-c^{-})c-(d-c)c^{-})(b-a)(d-c)(b^{-}-a^{-})-(b-a)(d^{-}-c^{-})$$

۶۲-۲- الگوریتم محاسبه‌ی پایین‌ترین جد مشترک

توضیح: ساده‌ترین راهی که به ذهن میرسد این است که به ازای هر پرسش(ui , vi) ، ابتدا راسی که در عمق پایین‌تری وجود دارد را تا جایی که عمق آن برابر با عمق راس دیگر شود بالا بیاوریم (یعنی u=father[u]) .

سپس تا زمانی که دو راس برابر نشدند در هر مرحله هر دو را یکی بالا میاوریم. از آنجایی که راس ریشه جد مشترک آن‌هاست این فرایند حتما متوقف می‌شود و راسی که در آن متوقف شدیم پایین‌ترین راسی‌است که جد مشترک ui و vi است. این الگوریتم اولیه از مرتبه‌ی O(n) برای هر پرسش عمل می‌کند و هزینه کل از مرتبه‌ی O(n*m) خواهد بود. (چرا؟) [3]

ایده‌ایی که برای بهبود عملکرد این الگوریتم به ذهن می‌رسد استفاده از روشی است که-i امین راس بالای هر راس دلخواه را در مرتبه‌ی خوبی در اختیار ما قرار بدهد. مقدار ancestor[v][i] را برابر با 2^i –امین راس بالای راس v در نظر می‌گیریم. برای تعیین مقادیر آرایه ancestor از روش برنامه‌ریزی پویا استفاده می‌کنیم:

- به ازای هر v و 0 = i مقدار ancestor[v][i]=father[v] است.
- به ازای هر i ناصفر

ancestor[v][i]=ancestor[ancestor[v][i-1]][i-1]
[i-1]است.

در قسمت اول الگوریتم ابتدا لازم بود که v و u را هم ارتفاع کنیم، می‌توانیم برای پیدا کردن-x امین راس بالای v با گام‌های به اندازه‌ی ۲ i بالا برویم و در O(lgn) مرحله راس مورد نظر را پیدا کنیم. (چرا؟) در قسمت بعدی برای پیدا کردن پایین‌ترین جد مشترک u و v می‌توانیم با جستجوی‌دودویی روی فاصله‌ی راس جواب و v جواب را در مرتبه‌ی O(lg 2 n) پیدا کنیم. می‌توانیم با کمی‌هوشمندی مرتبه‌ی پیدا کردن جواب برای هر پرسش را از O(lgn+lg 2 n)=O(lg 2 n) به O(lgn) کاهش دهیم) .راهنمایی: نمایش دودویی هر عدد کوچک‌تر از n حداکثر lgn رقم دارد.)

۶۳-۲- الگوریتم حذف معکوس

توضیح: الگوریتمی‌است که در یک گراف همبند، با یال وزن دار درخت پوشای کمینه را بدست می‌آورد. اگر گراف ناهمبند باشد این الگوریتم درخت پوشای کمینه را برای هر مولفه همبندی می‌یابد که در این صورت مجموعه این درخت‌های پوشای کمینه را یک جنگل پوشای کمینه گویند. [4]

این الگوریتم الگوریتمی‌حریصانه است که در هر لحظه بهترین انتخاب را انجام می‌دهد الگوریتم کراسکال با یک گراف خالی شروع می‌کند و یال‌ها را به آن اضافه می‌کند در حالیکه الگوریتم حذف معکوس با گراف اصلی شروع می‌کند و یال‌ها را از آن حذف می‌کند. الگوریتم به صورت زیر عمل می‌کند:

۱) با گراف G که لیستی از یال‌های E دارد شروع کن.

۲) Eرا به ترتیب نزولی وزن یال‌ها مرتب کن.

۳) برای هر یال چک کن که آیا حذف این یال گراف را ناهمبند می‌کند یا نه.

۴) اگر حذف کردن منجر به ناهمبندی گراف نمی‌شود یال را حذف کن

الگوریتم حذف معکوس این تضمین را می‌دهد که گراف همبند باقی بماند، چون تنها در صورتی یالی را حذف می‌کند که باعث ناهمبند شدن گراف نشود. هر یالی که حذف می‌شود قبل از حذف در دوری شرکت داشته‌است . [1]

از آنجایی که الگوریتم از یال با بیشترین وزن شروع به حذف کردن می‌کند، آن یال بزرگ‌ترین یال در دور مربوط به خود است. پس بنابر تعریف درخت پوشای کمینه، یال حذف شده جزء درخت پوشای کمینه نخواهد بود.

۶۴-۲- الگوریتم دسته بندی

توضیح: تعدادی از تکنیک‌های کلاس‌بندی آماری و یادگیری ماشینی در روش کلاس بندی در متن کاوی به کار برده می‌شود. برخی از این تکنیک‌ها عبارتند از:مدل‌های رگرسیون، کلاسه‌بندهای بیزی[1]،درخت‌های تصمیم‌گیری، کلاسه‌بندهای نزدیک‌ترین همسایه و ماشین‌های برداری و...که ما در این مقاله به بررسی و تحلیل بعضی از این تکنیک‌ها می‌پردازیم. [1]

KNN: k-Nearest Neighbor

مجموعه سندهای آموزشی است.

D = {D1, D2, …, Dn}

Q سند جدیدی است که قرار است مورد بررسی قرار گیرد.

TF ، term frequencyیک عبارت دریک سند است که به عنوان داده‌های آموزشی پیش پردازش شده است. [3]

V مقدار کسینوس شباهتQ و Di است. به طوری که(i=1,2,…,n)

تحلیل الگوریتم [KNN]

۱) وقتی یک سند به عنوان ورودی به برنامه داده می‌شود ابتداTF را برای همه کلمه‌های کلیدی در سند به دست می‌آوریم.

۲) سپس یک آرایه به سایز k درنظر می‌گیریم که ابتدا خالی است.این آرایه شامل id و نام و مقدار شباهت هرسند است. [2]

۳) یک شمارنده بامقدار اولیه صفر در نظر می‌گیریمcounter=0 .

i. برای هریک ازسندهایDi درمجموعه D اگر count<=k مقدارV بین Q وDiرا محاسبه می‌کنیم.) توجه داشته باشید که هنگام محاسبهV فقط کلمات کلیدی که درهردویQو Diدیده می‌شوند در نظرگرفته می‌شوند.) [2]

[1] Bayesian classifiers

ii. سپس شمارنده یکی اضافه می‌شود وv ونام سند در آرایه A[] نوشته می‌شود.

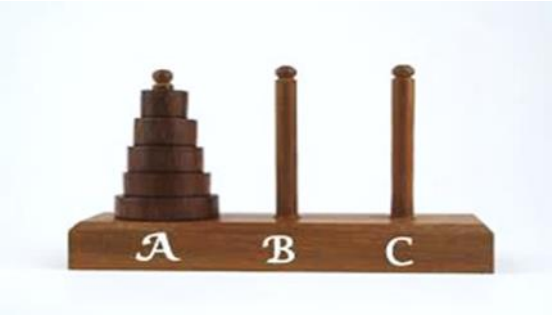
iii. درغیراین صورت یعنی اگرcount<=0 نباشداگرمجموعTFها در سند Di بیشتر از حداقل مقدار Vدرآرایه A[] باشد

iv. مقدار کسینوس شباهتV بین سندQو Di را محاسبه می‌کنیم..(باز هم توجه داشته باشید که هنگام محاسبهV فقط کلمات کلیدی که درهردوی Q و Diدیده می‌شوند در نظرگرفته می‌شوند.)

v. اگر مقدار جدید V محاسبه شده از حداقل مقدارV در آرایه A[] بیشتر بود سند را به همراه min(V) از آرایه A[] حذف می‌کنیم و سندQi و مقدار جدیدV را به A[] اضافه می‌کنیم. درغیر این صورت سند را رها می‌کنیم. [2]

۵) کلاسی را که بیشترین تعداد سند در آرایه A[] را دارد برمی‌گردانیم.

۶۵-۲- الگوریتم برج هانوی[۲۲]



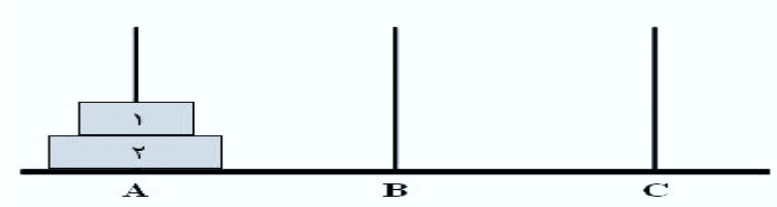
توضیح: سه میله‌ی - میله‌ی مبدأ(A) ، میله‌ی کمکی (B) و میله‌ی مقصد - (C) و تعدادی دیسک در میله‌ی مبدأ داریم. هدف انتقال تمام دیسک‌ها از این میله به میله‌ی مقصد با رعایت دو شرط زیر است: [1]

- در هر زمان فقط یک دیسک را می‌توان جابجا نمود.
- نباید در هیچ زمانی دیسکی بر روی دیسک با اندازه‌ی کوچکتر قرار بگیرد.

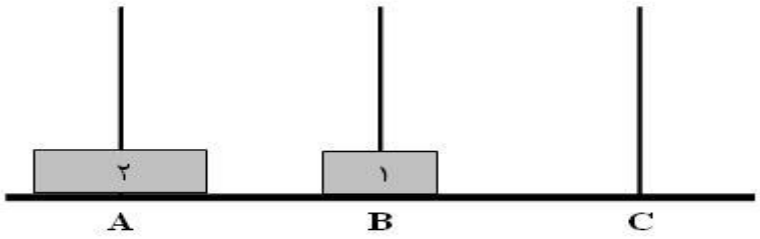
به طور حتم می‌توان با روش آزمون و خطا به نتیجه‌ی مطلوب رسید. اما هدف ما ارائه‌ی الگوریتمی‌برای انتقال دیسک‌ها با کمترین جابجایی ممکن است.

. به عنوان مثال، اگر n=2 باشد:

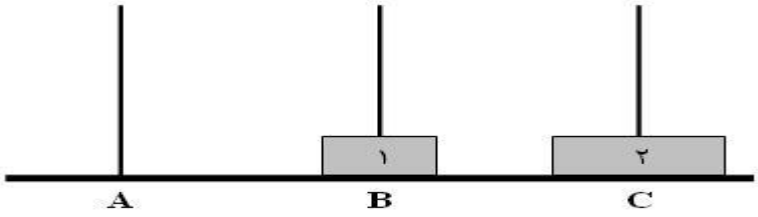
[2] Tower of Hanoi [2]



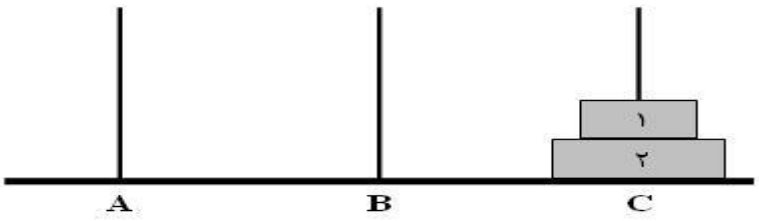
۱. دیسک 1را به میله‌ی Bمنتقل می‌کنیم (A→B)



۲. دیسک 2را به میله‌ی Cمنتقل می‌کنیم (A→C)



۳. دیسک 1را به میله‌ی Cمنتقل می‌کنیم (B→C)



توجه داشته باشید که بر اساس قانون اول، نمی‌توان به غیر از بالاترین دیسک هر میله، به دیسک دیگری از آن دسترسی پیدا کرد. [3]

۶۶-۲- الگوریتم حل فضای حالت برای دو ظرف

توضیح: دو ظرف به ظرفیت‌های ۶ و ۵ لیتری ۵،۶در ابتدا هر دو ظرف خالی هستند

۱) ظرف ۶ لیتری را از آب پر کنیم.

۲) ظرف ۵ لیتری را پر کنیم

۳) هر دو ظرف را خالی کنیم

۴) می‌توانیم آب ظرف‌ها را به یکدیگر بریزیم

۵) فقط در یک لحظه یک کار می‌توانیم انجام دهیم

مطلوب است که در ظرف ۶ لیتری ۳ لیتر آب باشد هدف(X, 3)

۶۷-۲- الگوریتم جاروبرقی

توضیح: در این مسئله عملکرد یک جاروبرقی هوشمند را مورد بررسی قرار می‌دهیم.

فرض می‌کنیم که دو اتاق وجود دارد که هر کدام از آن ها ممکن است شامل خاک باشد یا نباشد و عامل ممکن است در محیط ۱ یا ۲ باشد. عامل می‌تواند مستقیم برود و یا به چپ یا راست بپیچد. بنابراین هشت حالت ممکن، به عنوان عمل بعدی، وجود دارد. [2]

۱) جارو در محیط ۱ – هر دو محیط خاکی

۲) جارو در محیط ۱ – محیط ۱ خاکی

۳) جارو در محیط ۱ – محیط ۲ خاکی

۴) جارو در محیط ۱ – هیچکدام از محیط ها خاکی نیست

۵) جارو در محیط ۲ – هر دو محیط خاکی

۶) جارو در محیط ۲ – محیط ۱ خاکی

۷) جارو در محیط ۲ – محیط ۲ خاکی

۸) جارو در محیط ۲ – هیچکدام از محیط ها خاکی نیست

ابتدا تصور کنید که حسگرهای عامل به او اطلاعات کافی می‌دهند. (دنیا قابل دسترسی است.) و همچنین عامل می‌داند هر کدام از اعمالش چه تغییری در محیط ایجاد می‌کنند و سپس می‌تواند محاسبه کند با کدام وضعیت پس از انجام عمل رو به رو خواهد شد. این ساده‌ترین حالت مسئله است که به آن مسئله تک حالته می‌گویند. [3]

حالا تصور کنید که عامل تمام اثرهای عمل هایش را می‌داند، اما دسترسی به محیط محدود است. برای مثال، عامل هیچ حسگری ندارد. در این حالت فقط می‌داند که وضعیت اولیه اش یکی از اعضای مجموعه

(۱،۲،۳،۴،۵،۶،۷،۸) است. ممکن است فکر کنید که وضعیت عامل ناامید کننده است! اما در حقیقت اینقدر ها هم که به نظر می‌آید ناگوار نیست. [2]

زیرا عامل می‌داند که هر کدام از اعمالش چه اثری دارند. برای مثال می‌تواند محاسبه کند که عمل راست موجب می‌شود تا در یکی از حالات (۲،۴،۶،۸) باشد. بنابراین می‌تواند با انتخاب یک دنباله عملیاتی، به هدف برسد. [2]

به طور خلاصه هنگامی‌که محیط تماما قابل دسترسی نیست، عامل باید در مورد مجموعه حالت‌هایی که ممکن است هدف برسد، استدلال کند. به این نوع از مسئله مسئله چند حالته می‌گویند.

۶۸-۲- الگوریتم A*

توضیح: از جستجوی ابتدا بهترین استفاده می‌کند و کوتاه‌ترین مسیر را بین دو گره مبدا و مقصد داده شده به وسیله گره‌های دیگر می‌یابد.

این روش با ترکیب g(n) هزینه رسیدن به گره n و h(n) هیوریستیک یا تخمین هزینه رسیدن از گره n تا هدف گره‌ها را ارزیابی می‌کند:

از آنجایی که g(n) هزینه مسیر از گره شروع به گره n و h(n) هزینه تخمینی ارزان ترین مسیر از n{n} به هدف است، داریم :

بنابراین اگر به دنبال ارزان‌ترین راه حل هستیم، اولین کار معقول امتحان گره‌ای است که کمترین مقدار h(n) g(n)+را داراست. مشخص می‌شود که این راهبرد کاملاً معقولانه‌است. اگر تابع آروین h(n) شرایط خاصی داشته باشد. جستجوی A* هم کامل و هم بهینه‌است.

اگر A* همراه با الگوریتم جستجوی درخت استفاده شود، آنگاه تحلیل بهینگی آن بسیار ساده و سر راست می‌شود A* .بهینه‌است اگر h(n) یک آروین قابل قبول باشد.

یعنی h(n) هرگز هزینه رسیدن به هدف را بیش از اندازه واقعی برآورد نکند. این آروین‌ها ذاتاً بهینه هستند زیرا آنها فکر می‌کنند که هزینه راه حل مسئله کمتر از مقدار واقعی آن است.

از آنجا که g(n) هزینه رسیدن به n را به طور دقیق نشان می‌دهد، می‌توان بلافاصله نتیجه گرفت که f(n) هرگز هزینه واقعی راه حلی که ازn می‌گذرد را اضافه برآورد نمی‌کند.

۶۹-۲- الگوریتم جست‌جوی سطح‌اول

توضیح: جست‌وجوی اول‌سطح[۲۳] که به bfs مشهور استاین الگوریتم معمولا در گراف‌های وزن دار کاربرد خاصی ندارد و عمده نقش آن در گراف‌های بی‌وزن و برای پیدا کردن کوتاه‌ترین فاصله بین یک راس تا بقیه‌ی راس‌هااست.

این الگوریتم ابتدا یک راس ریشه مانند v را به عنوان شروع میگیرد. سپس راس‌ها را سطح بندی میکند .

سطح بندی به این صورت است که تمام راس‌های مجاور **v** را در سطح اول قرار میدهیم، حال برای سطح i+1 i+1، راس u که مجاور یکی از رئوس سطح i i است را در این سطح قرار میدهیم به شرطی که **u** در هیچ سطح دیگری نیامده باشد.

به عبارت دیگر راس‌ها را بر اساس کوتاه‌ترین فاصله از v سطح بندی می‌کنیم. حال برای پیمایش به ترتیب سطح، و در هر سطح به ترتیب دلخواه وارد راس‌ها می‌شویم. پس اولین زمانی که به هر راس می‌رسیم، با کمترین فاصله ممکن نسبت به v رسیده‌ایم.

با توجه به اینکه به هر راس حداکثر یکبار وارد می‌شویم در نتیجه هر یالی هم حداکثر دوبار به ازای دو سر آن شمرده می‌شود، پس در کل هزینه الگوریتم از O(n+e) O(n+e) است که **e** تعداد یال‌ها و n تعداد راس‌ها است.

از ویژگی های این الگوریتم این است که یال‌هایی که را که منجر به ایجاد دور می‌شوند را نمی‌رود در نتیجه اگر یال‌های رفته شده را کنار هم بگذاریم، تشکیل یک درخت می‌دهند.

راس‌های این درخت ریشه‌دار به چند سطح افراز شده‌اند که هر راسی در بالاترین سطح ممکن، یعنی در کمترین فاصله‌اش نسبت به ریشه آمده است.

لم: تمام یال‌های گراف اصلی که در درخت نیامده یا بین دو سطح مجاور یا درون یک سطح قرار دارند.

[2] Breadth First Search [3]

۷۰-۲- الگوریتم جستجوی IDA*[2]

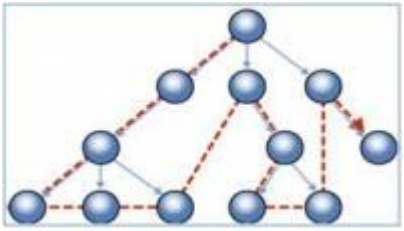
توضیح: می‌دانیم که جستجوی عمیق کننده تکراری تکنیکی مفید برای کاهش درخواست حافظه‌است حال می‌توانیم از این تکنیک استفاده نموده و هر بار یک جستجوی عمقی تا هزینه f-cost را جستجو کنیم. اگر به جواب نرسیدیم هزینه f-cost را افزایش داده و از اول به بسط فضای حالت می‌پردازیم.

نکته ۱: محدودیت هزینه در هر مرحله به گونه‌ای انتخاب می‌شود که در مراحل قبلی ثابت شده‌است که جوابی با هزینه کمتر از این مقدار وجود ندارد .

نکته ۲: می‌توان هزینه مرحله جدید را برابر با کمترین هزینه نودی که در مرحله قبلی بسط داده نشده قرار داد از آنجا که این روش به صورت عمقی است پس پیچیدگی فضایی آن در بدترین حالت b.d (S) است.

نکته ۳: پیچیدگی زمانی بستگی به تابع اکتشافی دارد.

نکته ۴: این روش نیز مشابه A* کامل و بهینه‌است



72- الگوریتم جست‌وجوی عمق‌اول:

توضیح: الگوریتم به این شکل است که ابتدا یک رأس مانند v را انتخاب می‌کنیم و آن را ریشه می‌نامیم. ریشه را علامت‌گذاری می‌کنیم. سپس یک رأس دل خواه علامت نخورده‌ی مجاور با v را انتخاب می‌کنیم و آن را u می‌نامیم.

u را یکی از بچه‌های v می‌کنیم، سپس u u را علامت می‌زنیم. حال همین الگوریتم را روی u از ابتدا اجرا می‌کنیم (یعنی یکی از همسایه‌های مجاور و علامت نخورده u u را انتخاب می‌کنیم و...).

الگوریتم گفته شده زمانی به بن‌بست می‌خورد که به ازای راسی مانندu ، تمام همسایه‌هایش علامت خورده باشند. در این صورت به راس پدر (رأسی که از آن وارد u شدیم) برمی‌گردیم و دوباره همین الگوریتم را از ادامه اجرا می‌کنیم (یعنی وارد پدر u می‌شویم و اگر همسایه‌ی علامت نخورده‌ای داشت وارد آن می‌شویم و اگر نداشت به رأس پدرش باز می‌گردیم).

[2] Intractive deeping A* [4]

برنامه زمانی متوقف می‌شود که به رأس v برگشته باشیم و تمام همسایه‌هایش علامت خورده باشند که در این صورت می‌گوییم الگوریتم پایان یافته است. دقت کنید که اگر گراف شما همبند نباشد، این جست‌وجو تنها رأس‌های مؤلفه ریشه را پیمایش می‌کند پس اگر برای پیمایش روی تمام رأس‌ها این الگوریتم را به ازای هر رأس علامت‌نخورده‌ای تکرار می‌کنیم.

پیمایش با انتخاب رأس r به عنوان ریشه آغاز می‌شود r .به عنوان یک رأس دیده شده برچسب می‌خورد. رأس دلخواه r1 از همسایگان r انتخاب شده و الگوریتم به صورت بازگشتی از r1 به عنوان ریشه ادامه می‌یابد. از این پس در هر مرحله وقتی در رأسی مانند v قرار گرفتیم که همه همسایگانش دیده شده‌اند، اجرای الگوریتم را برای آن رأس خاتمه می‌دهیم.

حال اگر بعد از اجرای الگوریتم با ریشه r1 همه همسایگان r برچسب خورده باشند، الگوریتم پایان می‌یابد. در غیر این صورت رأس دلخواه r2 از همسایگان r را که هنوز برچسب نخورده انتخاب می‌کنیم و جستجو را به صورت بازگشتی از r2 به عنوان ریشه ادامه می‌دهیم. این روند تا مادامیکه همه همسایگان r برچسب نخورده‌اند ادامه می‌یابد. [2]

البته پیمایش گراف برای تأمین هدفی صورت می‌گیرد. بر این اساس برای انعطاف پذیر ساختن الگوریتم در قبال کاربردهای مختلف، دو نوع عملیات preWORK و postWORK را به همراهِ بازدید از هر رأس یا یال انجام می‌دهیم، که preWORK در زمان برچسب خوردنِ رأسِ در حال بازدید، و postWORK بعد از بررسی هر یالِ خروجی از رأسِ در حال بازدید انجام خواهد شد. هر دوی این عملیات وابسته به هدفِ استفاده از الگوریتم، مشخص خواهند شد.

الگوریتم بازگشتی جستجوی اول عمق به صورت زیر است. آرایه یک بعدی Visited تعیین می‌کند آیا راسی قبلاً ملاقات شده است یا خیر. اگر راس vi ملاقات شود Visited[i] برابر با یک می‌شود.

۲-۷۱- جستجوی[۲۵]:SMA*

۱. می‌توان از تمام حافظه مورد دسترس خود استفاده کرد.

۲. تا جایی که حافظ اجازه می‌دهد از تولید حالات تکراری جلوگیری می‌کند.

۳. در صورتی کامل است که حافظه برای ذخیره کم عمق‌ترین مسیر راه حل وجود داشته باشد.

۴. در صورتی بهینه‌است که حافظه کافی برای ذخیره کم عمق‌ترین مسیر راه حل وجود داشته باشد.

[2] Simplified Memory Bounded A*

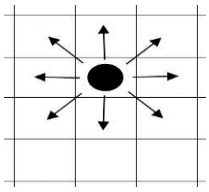
توضیح: زمانی که نیاز به تولید فرزند باشد ولی حافظه‌ای در اختیار نداشته باشیم نیاز به نوشتن مجدد به روی حافظه قبلی است برای انجام این امر الگوریتم یک گره را حذف می‌کند و فرزند جدید از حافظه آن استفاده خواهد کرد.

گره‌هایی که بدین طریق حذف می‌شوند گره‌های فراموش شده نام دارند. همیشه گره‌هایی برای حذف شدن انتخاب می‌شوند که هزینه آنها بالا است. برای جلوگیری از جستجوی مجدد زیر درخت‌هایی که از حافظه حذف شده‌اند، در گره پدر آنها اطلاعاتی در مورد کیفیت بهترین مسیر در زیر درخت فراموش شده نگهداری می‌شود. [4]

بدین طریق فقط زمانی زیر درخت‌ها دوباره تولید می‌شوند که ثابت شود تمام مسیرهای دیگر بدتر از مسیر فراموش شده باشد.

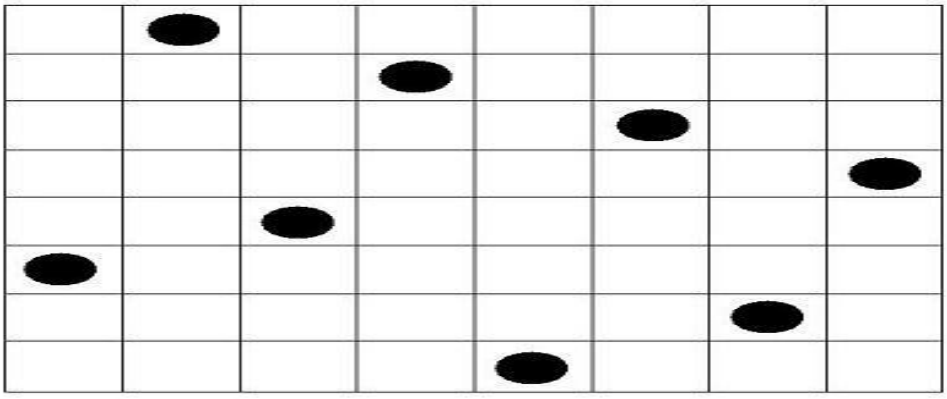
۷۲-۲- الگویتم هشت وزیر

توضیح: وزیر مهره‌ای از مهره‌های بازی شطرنج است که می‌تواند در تمامی‌هشت جهت به هر تعداد خانه – تا زمانی که مهره‌ای مانع نباشد– حرکت کند. اگر در این مسیرها مهره‌ای از حریف قرار گرفته باشد، آن مهره در معرض خطر حمله توسط وزیر قرار دارد؛ یا به اصطلاح وزیر آن مهره را تهدید می‌کند.



معمای n وزیر :

هدف از مسأله‌ی هشت وزیر، چیدن ۸ مهره‌ی وزیر روی یک صفحه‌ی شطرنج خالی است، به قسمی‌که هیچ مهره‌ای مهره‌های دیگر را تهدید نکند. به عبارت دیگر، ۸ وزیر باید به نحوی چیده شوند که هیچ‌کدام در یک سطر، یک ستون یا یک قطر قرار نداشته باشند. یک جواب مسأله می‌تواند به صورت زیر باشد.



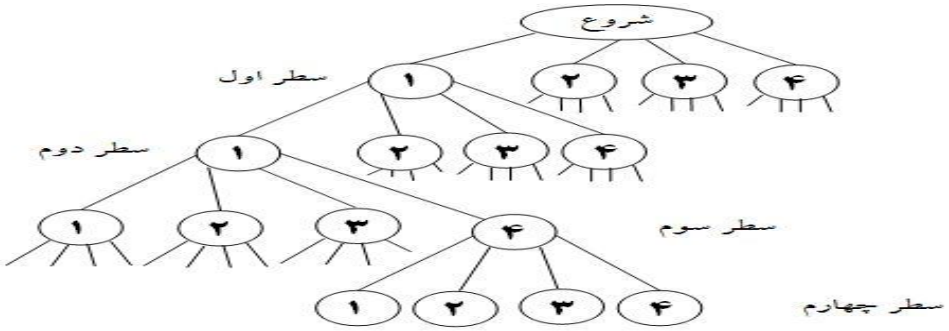
در حالت کلی به جای عدد ۸ از عدد طبیعی n استفاده شده و مسأله به ازای هر n بزرگتر یا مساوی ۴ مورد بررسی قرار می‌گیرد. به این ترتیب، هدف مسأله چیدن n مهره وزیر در یک صفحه شطرنج با ابعاد n x n است.

در یک صفحه‌ی n در n تعداد n2 خانه وجود دارد که از بین آنها n خانه برای قرار گرفتن n وزیر انتخاب می‌شود. در این انتخاب‌ها ترتیب اهمیتی ندارد .

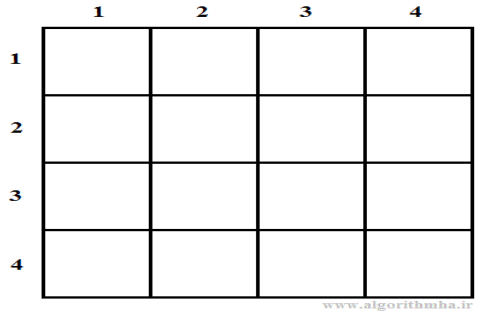
پس تعداد حالت‌های انتخاب n خانه برای چیدن n وزیر ترکیب n از n2 یا (C n2,n) است که حتی برای n های نه چندان بزرگ (نظیر ۸) عدد بزرگی به دست می‌آید.

در نتیجه بررسی تمامی‌حالات ممکن چینش مهره‌ها برای رسیدن به چیدمان صحیح به هیچ عنوان مقرون به صرفه نیست. از سوی دیگر به ازای هرn، تنها یک جواب منحصربفرد وجود ندارد. بنابراین اگر هدف مسأله یافتن تمامی‌جواب‌های ممکن باشد، استفاده از روش‌های هوشمند تکاملی یا الگوریتم‌های جستجوی تصادفی، لزوما ما را به نتیجه‌ی مطلوب نمی‌رساند[3]

روش عقبگرد : یکی از روش‌های حل این مسأله، استفاده از راهبرد عقبگرد است. در این روش تمام فضای مسأله به صورت یک درخت در نظر گرفته می‌شود که سطح iام شامل تمام انتخاب‌های مهره‌ی iام است. با توجه شرایط مسأله در هر سطر از صفحه‌ی شطرنج تنها یک مهره می‌تواند قرار بگیرد. به این ترتیب سطح شماره‌ی i، محل مهره i ام در سطر شماره‌ی i صفحه‌ی شطرنج را مشخص می‌کند. قسمتی از چنین درختی به ازای n=4 به این صورت خواهد شد.



پیمایش پیشوندی[۲۶]:چنین درختی تمام حالت‌های فضای مسأله را برای یافتن پاسخ جستجو می‌کند. اما حین پیمایش می‌توان از پیمایش گره‌های غیرامیدبخش صرف‌نظر کرد. گره‌های غیرامیدبخش گره‌هایی هستند که بر اساس تعاریف مسأله می‌توان مطمئن بود هرگز به جواب درست نمی‌رسد. به عنوان مثال، اگر با گره ۱ در سطح یک شروع کرده و به گره ۱ در سطح دو برویم، این گره امیدبخش نیست.



چرا که دو مهره اول در ستون شماره یک چیده شده‌اند و یکدیگر را تهدید می‌کنند. این اتفاق مستقل از این که مهره‌های بعدی در چه محل‌هایی قرار بگیرد مطمئنا به جواب صحیح ختم نمی‌شود.

در نتیجه از پیمایش فرزندان گره شماره‌ی ۱ در سطح دوم صرف نظر کرده و روال را با فرزند بعدی گره والد آن - یعنی گره شماره‌ی ۲ در سطح دوم - ادامه می‌دهیم. به همین ترتیب این گره نیز امیدبخش نیست. چرا که در چنین حالتی مهره در ستون شماره‌ی ۲ از سطر دوم قرار می‌گیرد که با وزیر سطر اول و ستون ۱ در یک قطر قرار دارند.

چنین پیمایشی معادل این است که از بالای صفحه‌ی شطرنج (سطر اول) شروع کرده و مهره را در ستون شماره‌ی ۱ قرار بدهیم. اگر چینش این مهره با مهره‌های بالاتر تهدیدی ایجاد نکند، چینش مهره‌ی بعدی در سطر بعدی را با همین روش ادامه می‌دهیم. اما اگر تهدیدی اتفاق بیفتد، مهره را یک خانه به سمت راست جابجا می‌کنیم.

[2] PreOrder [6]

اگر مهره به انتهای سطر رسیده باشد، آن مهره را برداشته و مهره‌ی سطر بالاتر را یک خانه به سمت راست جابجا می‌کنیم. اگر حین پیش‌روی سطرها به سمت پایین، به انتهای صفحه برسیم، یک جواب از مسأله تولید شده است. [3]

پیاده‌سازی مسأله: یکی از مهم‌ترین بخش‌های پیاده‌سازی مسأله‌ی ۸ وزیر، روش بررسی تهدید مهره‌ها است.

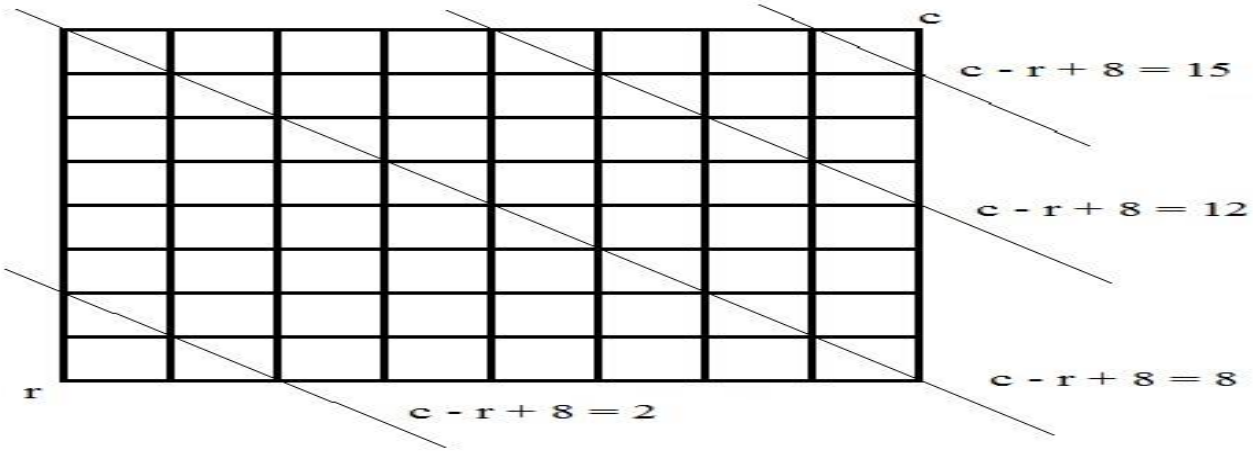
یک روش این است که یک آرایه دو بعدی n در n به عنوان صفحه‌ی شطرنج تعریف شود. با مشخص شدن محل هر مهره، تمام خانه‌هایی که توسط آن تهدید می‌شوند علامت‌گذاری می‌شود، تا بررسی غیرامن بودن آنها در مراحل بعدی آسان باشد. این روش در ابتدا بدون مشکل به نظر می‌رسد. اما سوای بحث مصرف حافظه، یک مشکل اساسی دیگر نیز وجود دارد. زمانی که تشخیص می‌دهیم فرزندان یک گره همه غیرامیدبخش هستند، از آن گره صرف نظر کرده و به گره بعدی فرزند والد مراجعه می‌کنیم. در نتیجه باید محل مهره در آن سطر جابجا شده و محل‌های غیرامنی که علامت‌گذاری شده بودند به حالت قبل از علامت‌گذاری بازگردند. اما مسأله اینجاست که محل‌های علامت‌گذاری شده پس از جابجایی لزوما امن نمی‌شوند. ممکن است همچنان توسط مهره دیگری تهدید وجود داشته باشد. تشخیص این مسأله که آیا آن محل به صورت امن مشخص شود یا نه، زمان مصرفی الگوریتم را دو چندان می‌کند.

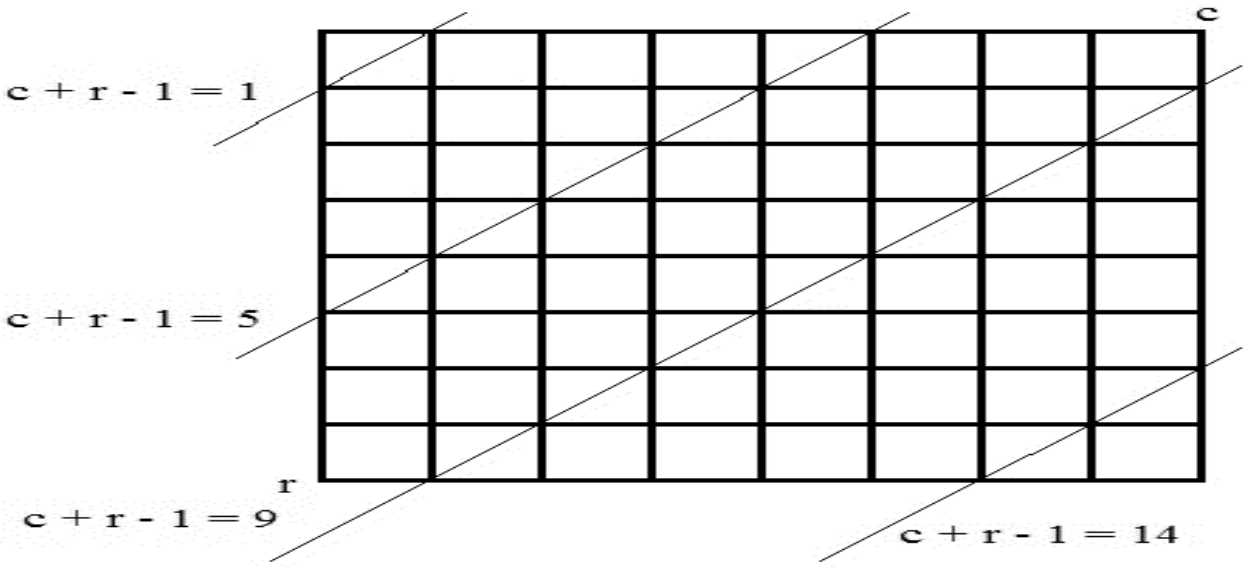
در روش دوم، تنها محل قرار گرفتن هر مهره به صورت شماره‌ی سطر و ستون ذخیره می‌شود. این ذخیره‌سازی می‌تواند در یک آرایه‌ی یک بعدی باشد. به این ترتیب که شماره‌ی اندیس آرایه، شماره‌ی سطر مهره را مشخص کرده و مقدار آن، شماره‌ی ستون مهره باشد. مثلا اگر مقدار اندیس شماره‌ی ۱ آرایه ۴ باشد، یعنی وزیر سطر اول در خانه‌ی چهارم قرار دارد. پس از این تعریف، با مشخص شدن محل یک مهره‌ی جدید، خانه‌هایی که ممکن است این خانه را تهدید کنند بررسی می‌شوند، تا امن بودن آن مشخص شود. مزیت این روش به روش قبلی – علاوه بر مصرف کمتر حافظه – رفع مشکل بررسی محل‌های علامت‌گذاری شده است.

برای بررسی این مسأله که آیا محل مهره‌ی جدید تهدیدی ایجاد می‌کند یا نه، به جای بررسی تک‌تک خانه‌های ستونی و قطری، از روابط ریاضی استفاده می‌شود. یافتن پاسخ این سوال که آیا در ستونی که مهره قرار دارد، قبلا مهره‌ای قرار داده شده است یا نه، کار چندان دشواری نیست. کافی است یک آرایه‌ی یک بعدی به طول n تعریف شده و هرگاه مهره‌ای در یک ستون مستقر شد، شماره‌ی ستون آن در آرایه به عنوان غیرامن علامتگذاری شود. در نتیجه مثلا اگر مهره‌ی جدید در ستون شماره‌ی ۳ قرار گرفت، کافی است اندیس شماره‌ی ۳ این آرایه برای اطمینان از امن بودن این ستون بررسی شود. [2]

در مورد دو قطر راست و چپ موضوع کمی‌پیچیده به نظر می‌رسد. اما با کمی‌دقت، یک رابطه‌ی ثابت ریاضی برای هر یک از قطرها وجود دارد. مثلا تفاضل شماره‌ی ستون از شماره‌ی سطر تمامی‌خانه‌های روی قطر اصلی

صفحه صفر است. یا مجموع شماره‌ی سطر و ستون تمامی‌خانه‌های قطر فرعی صفحه عدد ۹ است. به همین ترتیب سایر قطرها نیز رابطه‌ی ریاضی مشابه دارند. در نتیجه مثلا اگر مهره‌ی جدید در سطر سوم و ستون دوم قرار گرفته باشد، تنها کافی است در مهره‌های چیده شده‌ی قبلی دنبال مهره‌ای باشیم که تفاضل ستون از سطر آن عدد ۱، یا مجموع آنها عدد ۵ باشد. اگر چنین مهره‌ای یافت نشد، تهدیدی از طرف قطرها وجود ندارد. راه بهتر آن است که آرایه‌ای به طول n-1۲ برای دو قطر راست و چپ تعریف کرده و اگر مهره‌ای در یک قطر مستقر شد، آن قطر به عنوان قطر ناامن در آرایه علامتگذاری شود. در مرحله‌ی بعدی با قرار گرفتن هر مهره‌ی جدید، به جای بررسی تمام خانه‌های مهره‌های قبلی، این دو آرایه بررسی شوند. چنین آرایه‌هایی بر خلاف روش علامت‌گذاری قبلی، مشکل حذف علامت‌ها را ندارد (چرا؟).





نکته: با توجه به اینکه در قسمت پایین قطر اصلی تفاضل شماره‌ی ستون از شماره‌ی سطر عدد منفی می‌شود، برای حفظ استاندارد شروع از یک، آن را با عدد n جمع می‌زنند. همینطور در مورد مجموع شماره‌ی سطر و ستون، عدد یک کسر می‌شود[3] .

۷۳-۲- الگوریتم مسئله کوله پشتی با وزن ماکزیمم

توضیح: مسئله از این قرار است که یک کوله پشتی داریم که دارای وزن W هستش، یک سری اشیا هم با وزن معلوم $w(i)$ و ارزش معلوم $P(i)$ موجود هستش، هدف ما در این مسئله اینه که طوری کوله پشتی رو با این اشیا پر کنیم که مجموع ارزش اشیای انتخاب شده ماکزیمم باشد.

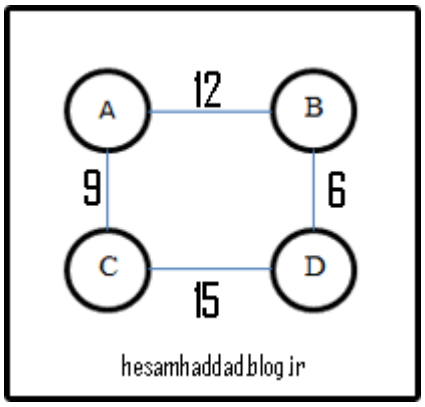
در کوله پشتی جزئی ما اجازه انتخاب کسری از شی رو هم داریم، خوب الگوریتم مسئله به این صورت هستش که باید اشیا رو بر اساس ارزش واحد وزن اونها مرتب کنیم به(صورت نزولی) و از ابتدا شروع کنیم و به سراغ اشیایی برویم که ارزش واحد وزن بیشتری دارند و تا جایی که کوله پشتی ما جا داره از اون شی برداریم و بعد اگه چیزی از کوله خالی موند به سراغ شی بعدی بریم و این کار رو تا جایی ادامه بدیم که کوله پشتی پر شود. (ارزش واحد وزن برای هر جسم از تقسیم ارزش کل جسم بر وزن همون جسم بدست میاد)

۷۴-۲- الگوریتم تپه‌نوردی

توضیح: تابع f به هر یک از اعضای مجموعه متناهی S یک عدد حقیقی نسبت می‌دهد. هدف ما یافتن عضوی از S است که بیش‌ترین (یا کم ترین) مقدار به آن نسبت داده شده است. همان گونه که گفته شد در این جا فرض بر این است که مقدار تابع f برای چندین عضو مختلف S بیشینه است و هدف ما یافتن تنها یکی از آن هاست. (در غیر این صورت معمولاً الگوریتم تپه نوردی، الگوریتم کار آمدی نیست.)

ابتدا یکی از اعضای مجموعه S را (به صورت تصادفی) انتخاب می‌کنیم و آن را A می‌نامیم. سپس در میان اعضایی از S که به A نزدیکند به دنبال پاسخ مناسب تری برای مسئله می‌گردیم. به بیانی دیگر تلاش می‌کنیم در میان اعضایی که به A نزدیکند عضوی بیابیم که مقدار تابع f برای آن بیش‌تر از $f(A)$ باشد. سپس این روند را ادامه می‌دهیم تا به یک بیشینه نسبی برای f دست یابیم. اگر الگوریتم گفته شده چندین بار تکرار شود می‌توان به یافتن پاسخی مطلوب امیدوار بود. (در هر بار اجرای الگوریتم نخستین عضو به صورت تصادفی انتخاب می‌شود.) بدی این الگوریتم این است که در یک تابع که مینیمم ویا ماگزیمم محلی زیادی دارد (مانند تابعAckley) به راحتی در ان گیر میفتد و قدرت خروج از آنجا را ندارد چون این الگوریتم فقط نقاط اطراف خود را میبیند که در لحظه حضور در یک ماگزیمم محلی این احساس را دارد که از همه نقاط بالاتر است در حالی که چنین نیست و ان در یک ماگزیمم محلی اسیر شده است. [1]

۷۵-۲- الگوریتم حریصانه



توضیح: برای پیدا کردن کوتاه‌ترین مسیر بین گره A و D اگر به صورت حریصانه عمل کنیم یال A-C از یال A-B دارای وزن کمتری می‌باشد پس یال A-C را در نظر می‌گیریم ولی در ادامه مجبوریم از تنها انتخاب یعنی C-D استفاده کنیم و به گره D برسیم. طول مسیر ۹ + ۱۵ یعنی ۲۴ می‌باشد اما در صورتی که در ابتدا یال A-Bرا انتخاب کرده بودیم با اینکه نسبت به A-C در مرحله اول انتخاب بدتری بود اما در ادامه مسیر بهینه‌تر و کوتاه‌تری را داشتیم. [1]

پس نمی‌توان از الگوریتم حریصانه در پیدا کردن کوتاه‌ترین میسر بین دو گره استفاده کرد برای اینکار می‌توان از الگوریتم‌هایی مثل Dijkstra استفاده کرد که به این مقاله مربوط نمی‌شود.

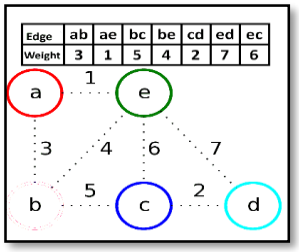
اما باید بدانیم که اگر در مسئله‌ای الگوریتم حریصانه درست باشد بی‌شک بهترین راه حل برای آن مسئله حریصانه است چون نسبت به بقیه الگوریتم‌ها دارای پیچیده گی زمانی کمتری می‌باشد.

در الگوریتم انتخاب بهینه فعالیت‌ها ابتدا کل بازه‌ها را بر اساس پایان به صورت صعودی مرتب می‌کنیم. حال اولین بازه را به مجموعه اضافه می‌کنیم و با یک حلقه بترتیب بازه‌های بعدی را چک می‌کنیم هر کدام که شروعش بعد از پایان آخرین بازه اضافه شده به مجموعه بود. به مجموعه اضافه می‌کنیم. و در پایان این مجموعه همان جواب مسئله است.

فقط نکته ای که وجود دارد این است که در این مسئله ما ID اولیه بازه‌ها را باید در خروجی بترتیب قرار گرفتن در مجموعه چاپ کنیم و با مرتب‌سازی‌ای که انجام می‌دهیم این ID از دست می‌رود پس نیاز به یک متغییر برای هر بازه داریم که در آن ID اولیه را ذخیره کنیم و بعد مرتب سازی این متغیر برای هر بازه تغییر نکند.

نکته دیگری وجود دارد این است که برای مرتب‌سازی روش‌های زیادی وجود دارد. در کتابخانه جدید C++ و Header file Algorithm تابعی به اسم Sort وجود دارد که در زمانی بسیار بهینه مرتب سازی را انجام می‌دهد. فقط این را در نظر بگیرید که ما برای کلاس‌ها و ساختمان هایمان باید عملگرهای کوچکتر و بزرگتر بودن را پیاده‌سازی کنیم در غیر این صورت مقایسه دو شی یک کلاس امکان پذیر نخواهد بود.

۷۶-۲- الگوریتم پریم



توضیح: در روش پریم ابتدا یک راس را انتخاب می‌کنیم. سپس کم وزن‌ترین یال‌های این راس را به عنوان یک یال درخت انتخاب می‌کنیم. حال بین یال های این دو راس کم‌وزن‌ترین یال را پیدا کرده و به عنوان عضوی از درخت انتخاب می‌کنیم. حال بین یال‌های این سه راس کم‌وزن‌ترین را انتخاب می‌کنیم. همینطور ادامه می‌دهیم تا درخت کامل شود. باید توجه داشت که یالی که هر بار اضافه می‌کنیم دور ایجاد نکند یا به عبارت دیگر یک سر آن در بین راس‌های غیر اتخاب شده باشد[1] .

۱. راس‌های گراف را به دو مجوعه V' و $V - V'$ (شامل همه راس‌های گراف به جز راس دلخواه v) افراز کن.

۲. تا زمانی که $V - V'$ شامل تمام راس‌ها نشده کارهای زیر را انجام بده

i . بین تمام یال‌هایی که بین مجوعه V' و $V - V'$ کم‌وزن‌ترین را انتخاب کن(مثلا یال (u, v) که u در V').

ii . راس v را از $V - V'$ حذف و به V' اضافه کن.

iii . یال (u, v) را به عنوان یالی از درخت پوشای کمینه انتخاب کن.

3 . پایان

۷۷-۲- الگوریتم تبرید شبیه‌سازی شده

توضیح: در روش شبیه سازی تبریدی(SA) ، هر نقطه s در فضای جستجو مشابه یک حالت از یک سیستم فیزیکی است، و تابع s (E) که باید کمینه شود، مشابه با انرژی داخلی سیستم در آن حالت است. در این روش، هدف انتقال سیستم از حالت اولیه دلخواه، به حالتی است که سیستم در آن کمترین انرژی را داشته باشد. برای حل یک مسئله بهینه‌سازی، الگوریتم SA ابتدا از یک جواب اولیه شروع می‌کند و سپس در یک حلقه تکرار به جواب‌های همسایه حرکت می‌کند .

اگر جواب همسایه بهتر از جواب فعلی باشد، الگوریتم آن را به عنوان جواب فعلی قرار می‌دهد (به آن حرکت می‌کند)، در غیر این صورت، الگوریتم آن جواب را با احتمال $\exp(-\Delta E/T)$ به عنوان جواب فعلی

می‌پذیرد. در این رابطه ΔE تفاوت بین تابع هدف جواب فعلی و جواب همسایه‌است و T یک پارامتر به نام دما است. در هر دما، چندین تکرار اجرا می‌شود و سپس دما به آرامی کاهش داده می‌شود[4] .

در گام‌های اولیه دما خیلی بالا قرار داده می‌شود تا احتمال بیشتری برای پذیرش جواب‌های بدتر وجود داشته باشد. با کاهش تدریجی دما، در گام‌های پایانی احتمال کمتری برای پذیرش جواب‌های بدتر وجود خواهد داشت و بنابراین الگوریتم به سمت یک جواب خوب همگرا می‌شود. الگوریتمSA یک الگوریتم غیرمقید می‌باشد که برای طراحی‌های سخت به کار می‌رود. [4]

در هر مرحله، الگوریتم تبرید شبیه‌سازی شده، چند حالت را در همسایگی حالت کنونی s در نظر می‌گیرد، و به طور احتمالی تصمیم می‌گیرد که سیستم را از حالت s منتقل کند یا در همین حالت باقی بماند. این احتمالات در نهایت سیستم را به حالت با انرژی کمتر میل می‌دهد.

۷۸-۲- الگوریتم کروسکال

توضیح: الگوریتم کروسکال، الگوریتم برای پیدا کردن درخت پوشای کمینه یک گراف وزن‌دار است. این الگوریتم بر خلاف الگوریتم پریم لزوما اجزایی که در حین اجرا جزء درخت پوشای کمینه تشخیص می‌دهد همبند نیستند و تنها تضمین می‌کند که در پایان این شرط برقرار است.

شرح: در این الگوریتم، یال‌ها را بر اساس وزن به صورت صعودی مرتب می‌کنیم. سپس از اولین یال شروع می‌کنیم و هر یالی که با یال‌هایی که قبلا انتخاب کردیم دور نمی‌سازد را انتخاب می‌کنیم. تا جایی که درخت تشکیل شود[1] .

اما چگونه متوجه شویم که یال جدید با یال‌های انتخابی قبلی دور نمی‌سازد؟ کافیست گرافی که تا کنون از یال‌های انتخابی تشکیل شده را به مؤلفه‌های همبندی تقسیم کنیم و اگر یال جدید هر دو سر آن در یک مؤلفه بود، یال جدید دور ایجاد می‌کند و در غیر اینصورت اضافه کردن یال جدید مشکلی ندارد .

هر راس متغیری همراه خود دارد که شماره مولفه‌ی همبندیش را نشان می‌دهد. در ابتدا هر راس، خود یک مؤلفه‌ی همبندیست. و وقتی یک یال بین دو مؤلفه ارتباط ایجاد می‌کند باید شماره مولفه‌ی همبندی هر دو گروه یکی شود .

در شکل یک مثال برای اجرای این الگوریتم را می‌بینید. توجه کنید یال‌های انتخاب شده تنها در آخر کار یک مجموعه‌ی همبند تشکیل می‌دهند ولی در الگوریتم پریم همیشه در طول ساخت درخت، مجموعه ما همبند بود .

۱) تمام یال‌ها را به ترتیب صعودی وزن مرتب کن

2) برای هر راس v یک مجموعه بساز.

3) یال (u , v) را انتخاب کن.

4) اگر مجموعه‌ی u و v یکی نیستند. کارهای زیر را انجام بده.

a . مجموعه‌ی آنها را ادغام کن.

b . یال (u , v) را به عنوان یالی از درخت پوشای کمینه بردار.

5) اگر هنوز n-1 یال انتخاب نشده به شماره ۳ برو.

6) پایان

۷۹-۲- الگوریتمی‌بنویسید که که اعضای ۲ مجموعه را از ورودی گرفته مجموعه‌ی اشتراک و مجموعه اجتماع این ۲ مجموعه را محاسبه و چاپ نماید

توضیح:

1) اعضاء مجموعه اول را در آرایه a دریافت کند.

2) اعضاء مجموعه دوم را در آرایه b دریافت کند.

3) تعداد اعضاء مجموعه اول را در n قرار دهد.

4) تعداد اعضاء مجموعه دوم را در m قرار دهد.

5) مقدار ماکزیمم دو مقدار m و n را در max قرار دهد.

6) مقدار مینیمم دو مقدار m و n را در min قرار دهد.

7) آرایه c را به طول max خانه ایجاد کند.

8) آرایه d را به طول min خانه ایجاد کند.

9) در متغیر i مقدار ۱ را قرار دهد.

10) در متغیر x مقدار ۱ را قرار دهد.

11) در متغیر y مقدار ۱ را قرار دهد.

12) اگر مقدار i از n بزرگتر بود به مرحله ۲۱ برو.

13) اگر مشابه مقدار a[i] در داخل آرایه c بود به مرحله ۱۹ برو.

14) مقدار خانه c[x] را برابر a[i] قرار بده.

15) مقدار x را یک واحد افزایش بده.

16) اگر مشابه مقدار a[i] در داخل آرایه b نبود به مرحله ۱۹ برو.

17) مقدار خانه d[y] را برابر a[i] قرار بده.

18) مقدار y را یک واحد افزایش بده.

19) مقدار i را یک واحد افزایش بده.

20) به مرحله ۱۲ برو.

21) در متغیر j مقدار ۱ را قرار دهد.

22) اگر مقدار j از m بزرگتر بود به مرحله ۲۸ برو.

23) اگر مشابه مقدار b[j] در داخل آرایه c بود به مرحله ۲۶ برو.

24) مقدار خانه c [x] را برابر b[j] قرار بده.

25) مقدار x را یک واحد افزایش بده.

26) مقدار j را یک واحد افزایش بده.

27) به مرحله ۲۲ برو.

28) اعضاء آرایه c به طول x خانه را به عنوان اجتماع دو مجموعه نمایش بده.

29) اعضاء آرایه d به طول y خانه را به عنوان اشتراک دو مجموعه نمایش بده.

30) پایان

۸۰-۲- الگوریتم ضرب ماتریس‌ها

توضیح: ماتریس‌ها به طور ساده، همانند آرایه‌های دوبعدی اند. یک ماتریس n×m n×m یعنی ماتریسی که n nسطر و m m ستون دارد (تعداد سطرها و ستون‌ها می‌تواند برابر باشد).

برای ضرب کردن دو ماتریس، باید تعداد ستون‌های اولی با تعداد سطرهای دومی‌برابر باشد. ضرب‌کردن یک ماتریس n×m n×m در یک ماتریس m×p m×p به اندازه‌ی n×m×p n×m×p هزینه (زمان) می‌برد و خروجی‌اش یک ماتریس n×p n×p است.

اما اگر بیش از دو ماتریس داشته باشیم، این که به چه ترتیبی این ضرب‌ها را انجام بدهیم (یا چگونه بین ماتریس‌ها پرانتزگذاری کنیم) در هزینه‌ی نهایی تاثیر دارد (دقت کنید که جواب یکسان است، فقط هزینه‌ای که ما صرف می‌کنیم فرق می‌کند.)

برای مثال:

فرض کنید، سه ماتریس به ابعاد (۲،۳) و (۳،۴) و (۴،۵) داشته باشیم. اگر ابتدا اولی و دومی‌و سپس نتیجه را با سومی‌ضرب کنیم، هزینه‌ی ما ۶۴ واحد می‌شود. اما اگر ابتدا دومی‌و سومی‌را ضرب کرده و سپس اولی را با نتیجه ضرب کنید، ۹۰ واحد باید هزینه کنید .

$2\times3\times4+2\times4\times5=64$ $2\times3\times4+2\times4\times5=64$

$3\times4\times5+2\times3\times5=90$

می‌توانید ببینید که تعداد حالت‌های مختلف انجام این پرانتزگذاری بسیار زیاد است. در واقع این مقدار برابر $(k-1)\ !\ \ (k-1)\ !$ است. اما در مجموع، برای حل بسیاری از این حالت‌های مختلف، باید مقادیر تکراری زیادی حساب کنیم.

همانطور که در ابتدا گفته شد، در این مسئله فقط باید مشخص کنیم که پرانتزگذاری بین ماتریس‌ها باید چگونه باشد. و اگر دقت کنید، در صورتی که تمام ماتریس‌های ii ام تا jj ام در یک پرانتز کلی قرار داشته باشند، فرقی نمی‌کند که چگونه عملیات ضرب را در این تکه انجام دهید، چون در هر صورت، ماتریس پایانی این مجموعه یکسان است (درنتیجه ابعاد آن نیز یکسان است) .

پس بهتر است این تکه را هم به بهترین روش (با کمترین هزینه) حل کنیم. این مسئله خود شبیه مسئله‌ی اصلی است با این تفاوت که دنباله‌ی ماتریس‌هایمان فقط ماتریس‌های ii ام تا jj ام اند.

با داشتن این ایده در ذهن بیایید به سراغ سوال اصلی برویم. می‌خواهیم سوال اصلی را به یک سری از زیرسوال‌هایی که در بالا تعریف کردیم، تبدیل کنیم. با کمی‌بررسی بیشتر، متوجه می‌شویم که اگر آخرین ضربی که باید انجام شود را تعیین کنیم، باید چیزی به این شکل باشد :

$(A_1, A_2, \ldots, A_i) \times (A_{i+1}, A_{i+2}, \ldots, A_k)$
$(A1, A2, \ldots, Ai) \times (Ai+1, Ai+2, \ldots, Ak)$

پس راه حل به دست آمده را می‌توان به زبان ریاضی به شکل زیر تعریف کرد :

(فرض کنید اندازه‌ی بعد اول یا همان سطرهای ماتریس اول تا k k ام را a 1 a1 تا a k ak و بعد دوم یا ستون‌های ماتریس آخر را a k+1 ak+1 بنامیم. می‌دانیم که باید برای مثال اندازه‌ی بعد دوم ماتریس i i ام، برابر بعد اول ماتریس i+1 i+1 باشد، وگرنه ضرب ممکن نیست.

مقدار d i,j di,j را برابر کمترین هزینه‌ی ضرب کردن ماتریس i i ام تا j j ام در نظر بگیرید. پس جواب مسأله برابر مقدار d ۱,k d۱,k است.

مقدار اولیه d i,i di,i :برابر صفر است، چون هیچ نیاز به انجام هیچ ضربی نیست.

به روز رسانی: مقدار d i,j di,j با شرط j>i j>i بسته به اینکه کدام ضرب در انتها انجام شود، به صورت روبرو تعریف می‌شود :

d i,j =min i≤m<j (d i,m +d m+1,j +a i ×a m+1 ×a j+1)

۸۱-۲- الگوریتم استراسن

توضیح

n هستند و n*n هر کدام یک ماتریس A و B را حساب می‌کنیم.

که C=AB فرض می‌کنیم می خواهیم حاصل ضرب

C=AB ومعادله تقسیم می‌کنیم n/2*n/2 به ۴ ماتریس A, B, C دقیقا توانی از ۲ است.

ما هر یک از ماتریس‌های را به صورت زیر بازنویسی می‌کنیم.

$$(28.8)\quad \begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}.$$

$$(28.9)\quad r = ae + bg.$$
$$(28.10)\quad s = af + bh.$$
$$(28.11)\quad t = ce + dg.$$
$$(28.12)\quad u = cf + dh.$$

n/2*n/2 و جمع حاصل ضرب‌های n/2*n/2 تا از ضرب های ماتریس های ۲ هر یک از معادلات بالا

به صورت زیر است:

n*n دستور بازگشتی زیر برای پیچیدگی زمانی ۲ ماتریس به شکل زیر است.

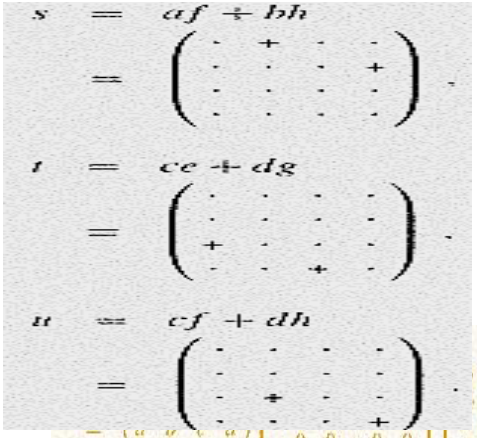
$$(28.13)\ T(n) = 8T(n/2) + \Theta(n^2)\ .$$

بنابراین این روش سریع‌تر از یک روش عادی نیست$\Theta= (n) T.$ $(n3)$ متاسفانه پیچیدگی زمانی

جمع‌ها $\Theta(n\ 2)$ و $n/2*n/2$ استراسن یک روش بازگشتی متفاوت که تنها به ۷ عمل بازگشتی ماتریس‌های و تفریق‌های اسکالر نیاز دارد.

$$\begin{aligned}(28.14)\ T(n) &= 7T(n/2) + \Theta(n^2)\\ &= \Theta(n^{\lg 7})\\ &= O(n^{2.81})\ .\end{aligned}$$

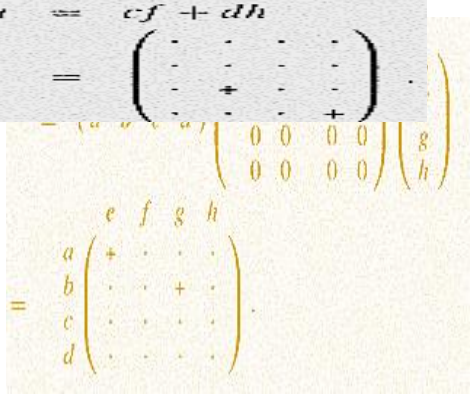
روش استراسن ۴ مرحله دارد:



1) $n/2*n/2$به زیر ماتریس‌های A و $B1-$ تقسیم کردن ماتریس‌های ورودی

2) $A\ 1B1, A2B2, \ldots\ldots A7B7$ استفاده از جمع و تفریق اسکالر و محاسبه ۱۴ ماتریس

3) $Pi=AiBi \quad for\ i=1,2\ldots7-$ محاسبه حاصل ضرب باز گشتی



4) Piی ونتیجه ماتریس‌ها C با جمع وتفریق ترکیب‌های متفاوت ماتریس‌هاr, s, t, u محاسبه ماتریس‌های

تعیین کردن حاصل ضرب‌های زیر ماتریس‌ها:

دقیقا واضح نیست که چگونه روش استراسن ضرب زیر ماتریس‌ها را که کلید کار الگوریتمش هست را می‌سازد. ما یک روش کشف شده احتمالی را باز سازی می‌کنیم.

می‌تواند به این فرم نوشته شود Pi.حدس می‌زنیم که حاصل ضرب ماتریس

$$\begin{aligned}(28.15)\ P_i &= A_i B_i\\ &= (\alpha_{i1}a + \alpha_{i2}b + \alpha_{i3}c + \alpha_{i4}d)\cdot(\beta_{i1}e + \beta_{i2}f + \beta_{i3}g + \beta_{i4}h)\ ,\end{aligned}$$

تعلق دارند. ما حدس میزنیم که هر یک از حاصل ضرب‌ها با جمع وتفریق $\{-1,0,+1\}$ به مجموعه Pi که ضرایب حساب می‌شود وسپس دو نتیجه با هم ضرب می‌شود .

A , B تعدادی از زیرماتریس‌های اگر ما ۴ حاصل ضرب در این روش تشکیل می‌دهیم و سپس ما می‌توانیم از این روش بازگشتی بدون فرض جا به را در سمت چپ A جایی ضرب استفاده کنیم.

هر یک از حاصل ضرب‌ها همه زیر ماتریس‌های را در سمت راست دارد. این خاصیت ضروری است برای کاربرد بازگشتی این روش B وزیر ماتریس‌های از آنجایی که ماتریس حاصل ضرب جابه جایی پذیر نیست.

برای اسان شدن از ماتریس ۴*۴ استفاده می‌کنیم تا نشان دهم ترکیب‌های خطی زیر ماتریس‌ها که هر ضرب ترکیب (که به صورت زیر باز نویسی می‌کنیم.۲۸-۱۵ در معادله) A با B می‌کند یکی از زیر ماتریس‌های به جای ۰از (.)و به جای۱+از (+)و به جای۱-از(-) استفاده می‌کنیم.

شروع می‌کنیم که که محاسبه S ما تحقیق‌مان را برای یک الگوریتم ضرب ماتریس سریع با مشاهده زیرماتریس حساب می‌شود.

با استفاده از ضرب یک ماتریس P1 , P2که S=P1+P2 می‌شود.

۸۲-۲- الگوریتم مؤلفه قوی مبتنی بر مسیر

توضیح: فرض کنید گراف G داده شده است. در این الگوریتم رأس‌های گراف را از ۱ تا n و مولفه‌های قویا همبند را با شروع از n+۱ شماره‌گذاری می‌کنیم .

همچنین در این الگوریتم گراف H را که در ابتدا همان گراف G می‌باشد در نظر می‌گیریم و در این گراف مسیر Pرا به صورت زیر می‌سازیم: ابتدا یک رأس دلخواه از گراف را در نظر می‌گیریم و از این رأس در جهت کاملاً دلخواه حرکت می‌کنیم.

تا یکی از دو حالت زیر رخ دهد:

1) اگر به رأسی رسیدیم که قبلاً در مسیر P وجود داشت، تمام گره‌های بین این رأس که تا کنون پیموده شده است را هم در گراف H و هم در مسیر p منقبض می‌کنیم. یعنی همه آن‌ها را به عنوان یک رأس در نظر می‌گیریم.

2) اگر به رأسی رسیدیم که دیگر نتوانستیم مسیر P را ادامه دهیم، گره آخر مسیر P را هم در گراف H وهم در مسیر P حذف کرده و آن را به عنوان یک مولفه قویا همبند معرفی می‌کنیم.

این الگوریتم از الگوریتم جستجوی عمق اول و همچنین دو پشته برای نمایش مسیر P استفاده می‌کند. پشته S که شامل دنباله رأس‌های مسیر P است و پشته B که شامل کران‌های رأس‌های منقبض شده است.

در این الگوریتم آرایه I نیز وجود دارد که هم اندیس پشته S را و هم شماره مولفه قویا همبند را ذخیره می‌کند. به عبارت دقیق تر این ارایه به صورت زیر تعریف می‌شود:

1) I[v] = ۰ هرگاه گره v در مسیر P نباشد.

2) I[v]=j هرگاه گره v اکنون در مسیر P باشد و S[j]=v.

3) I[v]=c هرگاه مولفه قویا همبند شامل v حذف شده و با عدد c شماره گذاری شده باشد.

۸۳-۲- الگوریتم گابو

توضیح: الگوریتم گابو همانند الگوریتم مولفه‌های همبند قوی تارجان، تنها یک جستجوی عمق اول در گراف داده شده انجام می‌دهد، و از یک پشته برای نگهداری گره‌هایی که به مولفه‌ای اختصاص داده نشده‌اند استفاده می‌کند، و بعد از اختصاص دادن آخرین گره به مولفه همبندی، این گره‌ها را به یک مولفه جدید منتقل می‌کند.

الگوریتم تارجان از یک آرایه با اندیس گره‌ها شامل اعداد پیش‌ترتیب استفاده می‌کند، که به ترتیب دیده شدن در جستجوی عمق اول مقداردهی شده‌اند.

از آرایه پیش‌ترتیبی برای دانستن اینکه چه زمانی مولفه همبندی جدید ایجا شود استفاده می‌گردد. الگوریتم گابو به جای استفاده از آرایه، از پشته دیگری به این منظور استفاده می‌کند.

الگوریتم گابو با پشتیبانی دو پشته S و P یک جستجوی عمق اول بر روی گراف داده شده G انجام می‌دهد. پشته S تمامی‌گره‌هایی را تاکنون به یک مولفه همبندی قوی اختصاص داده نشده‌اند، شامل می‌شود.

گره‌ها در این پشته به ترتیبی که جستجوی عمق اول به آن می‌رسد قرار دارند. پشته P شامل گره‌هایی است که هنوز اختصاص آن‌ها به مولفه‌های همبندی

قوی متفاوت از هم، تعیین نشده است. همچنین در الگوریتم از یک شمارنده C که تعداد گره‌های مشاهده شده تاکنون است، برای محاسبه اعداد پیش‌ترتیب گره‌ها استفاده می‌شود.

هنگامی‌که جستجوی عمق اول به یک گره v می‌رسد، الگوریتم مراحل زیر را به ترتیب انجام می‌دهد:

1) عدد پیش‌ترتیب v را برابر C قرار می‌دهد، و مقدار C را یکی افزایش می‌دهد.

2) گره v را در پشته S و همینطور P قرار می‌دهد.

3) برای هر یال از گره v به گره مجاورw :

i. اگر عدد پیش‌ترتیب w هنوز مقداردهی نشده، به طور بازگشتی جستجو را روی w انجام می‌دهد؛

ii . در غیر اینصورت، اگر w هنوز به یک مولفه همبندی قوی اختصاص داده نشده:

iii . تا زمانی که عدد پیش‌ترتیب عنصر بالای پشته P، بزرگتر اکید عدد پیش‌ترتیب w است، عنصر بالای P را خارج می‌کند.

4) اگر v عنصر بالای P بود:

i . عناصر بالای S را تا زمانی که v خارج شود، خارج کرده و این عناصر را به یک مولفه همبندی اختصاص می‌دهد.

ii . گره v را از P خارج می‌کند.

الگوریتم کلی شامل یک حلقه روی گره‌های گراف G است، که این جستجوی بازگشتی را برای گره‌هایی که هنوز عدد پیش‌ترتیب آن‌ها مقداردهی نشده است، صدا می‌زند.

۸۴-۲- الگوریتم دکسترا

روند الگوریتم دایکسترا مطابق زیر می‌باشد:

1) انتخاب راس مبدا

2) مجموعه S ، شامل رئوس گراف، معین می‌شود. در شروع، این مجموعه تهی بوده و با پیشرفت الگوریتم، این مجموعه رئوسی که کوتاه ترین مسیر به آن‌ها یافت شده است را در بر می‌گیرد.

3) راس مبدا با اندیس صفر را در داخل S قرار می‌دهد.

4) برای رئوس خارج از S ، اندیسی معادل، طول یال + اندیس راس قبلی، در نظر می‌گیرد. اگر راس خارج از مجموعه دارای اندیس باشد، اندیس جدید کمترین مقدار از بین اندیس قبلی و طول یال + اندیس راس قبل، می‌باشد.

5) از رئوس خارج مجموعه، راسی با کمترین اندیس انتخاب شده و به مجموعه S اضافه می‌گردد.

6) این کار را دوباره از مرحله ۴ ادامه داده تا راس مقصد وارد مجموعه S شود.

7) در پایان اگر راس مقصد دارای اندیس باشد، اندیس آن نشان دهنده مسافت بین مبدا و مقصد می‌باشد. در غیر این صورت هیچ مسیری بین مبدا و مقصد موجود نمی‌باشد.

توضیح: همچنین برای پیدا کردن مسیر می‌توان اندیس دیگری برای هر راس در نظر گرفت که نشان دهنده راس قبلی در مسیر طی شده باشد. بدین ترتیب پس از پایان اجرای الگوریتم، با دنبال کردن رئوس قبلی از مقصد به مبدا، کوتاه‌ترین مسیر بین دو نقطه نیز یافت می‌شود.

در حین اجرای الگوریتم دو چیز به طور ضمنی نگهداری می‌شود. یکی مجموعه از رأس‌هایی که وزن کوتاه‌ترین مسیر از مبدأ تا آن‌ها مشخص شده و دیگری دنباله که برای هر رأس، مقدار برابر وزن کوتاه‌ترین مسیر از مبدأ تا است به شرطی که تمام رأس‌های این مسیر به جز از رئوس داخل باشند. در ابتدا تهی و مقادیر برای همه رئوس به غیر از مبدأ بی‌نهایت است و مقدار آن برای مبدأ صفر گذاشته می‌شود.

الگوریتم در هر مرحله رأسی خارج را که برای آن کمترین است انتخاب و به مجموعه اضافه می‌کند و سپس مقادیر را برای رئوس همسایه آن رأس به‌روز می‌نماید. در صورتی که نیاز به تشکیل درخت کوتاه‌ترین مسیر باشد، الگوریتم می‌بایست دنباله را که پدر رأس در درخت کوتاه‌ترین مسیر است، به همراه دنباله به‌روز کند.

در پیاده‌سازی، برای اینکه مشخص کنیم چه رئوسی در مجموعه هستند، در هر مرحله رأسِ وارد شده به را برچسب می‌زنیم.

۸۵-۲- الگوریتم میانگین–خطی درخت پوشای کمینه

توضیح: حرکت بروکا شامل هر تکرار الگوریتم مبتنی بر الگوریتم بروفکا را یک گام در نظر می‌گیریم.

ورودی: یک گراف که راس تنها ندارد

1) یک گراف منقبض به وسیله‌ی جایگزینی مولفه‌های ساخته شده با یال‌های مرحله یک با یک راس بساز
2) تمام راس‌های تنها، طوقه‌ها، یال‌های تکراری غیر کمینه را حذف کن
3) خروجی: یال‌های انتخاب شده در مرحله یک و گراف منقبض شده‌ی باقی‌مانده

یالهای F-heavy و:F-light

در هر تقسیم یال‌هایی با خواصی که آن‌ها را از بودن در درخت پوشای کمینه محروم می‌کند حذف می‌شوند. این یال‌ها را F-heavy می‌نامیم که این گونه اند: فرض کنید F یک جنگل پوشای کمینه از گراف H است، یک F-heavy یالی است که راس‌های u و v را به هم دیگر متصل می‌کند که وزنش از وزن بیشترین یال مسیر u و v در F بیشتر است .

هر یالی که F-heavy نیست F-light نام دارد. اگر H یک زیرگراف از G باشد، هیچ F-heavy نمی‌تواند در درخت کمینه باشد. برای گراف G می‌توان F-heavy ها را در زمان مورد نظر خطی محاسبه کرد.

الگوریتم:

1) یک گراف که راس تنها ندارد بگیر
2) اگر گراف تهی بود یک جنگل تهی بده
3) یک گراف منقبض با اجرای ۲ مرحله پی در پی بروفکا بساز
4) یک زیر گراف با انتخاب یال ها به احتمال (۱/۲) بساز و درخت پوشای کمینه ی آن را بیاب
5) با استفاده از linear time minimum spanning tree verification algorithmتمام F-heavy ها را حذف کن
6) به صورت بازگشتی درخت پوشای کمینه را برای G' بدست بیاور
7) درخت پوشای کمینه ی G' و یال های منتخب بروفکا را چاپ کن

۸۶-۲- الگوریتم انتخاب بهینه فعالیت‌ها

توضیح: مسئله انتخاب فعالیت‌ها از مسائل بهینه‌سازی ریاضی است که می‌توان برای آن الگوریتمی به روش حریصانه تولید کرد. برای نمونه فرض کنید n سخنران خود را به عنوان ورودی مسئله اعلام کرده‌اند. هدف انتخاب بیشترین تعداد سخنران به گونه‌ای است که هیچ دو سخنرانی با هم اشتراک بازه زمانی نداشته باشند.

جهت عدم تداخل در زمان شروع یا پایان می‌توان بدون کم‌شدن از کلیت مسئله فرض کرد که پایان بازه سخنرانی‌ها باز است یعنی به صورت (...} است.

ورودی‌ها:

- siشروع فعالیت i ام
- Fiپاین فعالیت i ام
- nتعداد فعالیت‌ها
- هدف: انتخاب بیشترین تعداد فعالیت به قسمی‌که اشتراک بازه زمانی نداشته باشند.

توضیح الگوریتم:

1) در ابتدا فعالیت‌ها را بر اساس زمان پایان آنها مرتب کرده و سپس اولین فعالیت (زودترین زمان پایان) را انتخاب می‌کنیم.

2) به ازای i از ۲تا n مرحله ۳ را تکرار می‌کنیم

3) اگر فعالیتi ام با آخرین فعالیت انتخاب شده تداخل بازه زمانی نداشت آن را انتخاب می‌کنیم.

۸۷-۲- الگوریتم امید ریاضی–بیشینه کردن

توضیح: الگوریتم امید ریاضی–بیشینه‌سازی (EM) یک روش تکرارشوندهٔ[۲۷] است که به دنبال یافتن برآوردی با بیشترین درست نمایی برای پارامترهای یک توزیع پارامتری است. این الگوریتم روش متداول برای زمانهایی است که برخی از متغیرهای تصادفی پنهان هستند.

- فرض کنید که مشاهدات x 1 , x 2 ,..., x n را با d نمایش دهیم، متغیرهای پنهان , h 1 h 2 ,..., h n را با h و همه ی پارامترهای توزیع را نیز با θ در این صورت لگاریتم درست نمایی کل داده‌ها (پنهان و نمایان=مشاهدات) برابر خواهد بود با:

$$\mathcal{L}(\theta) = \log p(d;\theta) = \log \sum_{h} p(d,h;\theta)$$

•از آنجا که لگاریتم تابع اکیداً صعودی است، می‌توان لگاریتم درست نمایی کل داده ها را نسبت به θ بیشینه کرد. هرچند، آرگومان لگاریتم یک مجموع است و نمی‌توان به سادگی پاسخ تحلیلی برای θ یافت. از این رو، الگوریتم ب–ا ترفندی را برای بیشینه کردن حد پایین لگاریتم درست نمایی بکار می‌برد. این حد پایین از نابرابری جنسن بدست می‌آید. بر اساس نابرابری جنسن که از مجموعه مقعر بودن تابع لگاریتم استفاده می‌کند برای هر دسته k تایی از t i ها و w i ها اگر t i > 0 و خواهیم داشت.

$$\sum_{i=1}^{k} w_i \log t_i \leq \log \sum_{i=1}^{k} w_i t_i$$

اکنون L را به صورت زیر باز می‌نویسیم

[2] iterative [7]

$$\mathcal{L}(\theta) = \log \sum_h q(h) \frac{p(d,h;\theta)}{q(h)} \geq \sum_h q(h) \log \frac{p(d,h;\theta)}{q(h)} = \mathcal{J}(q,\theta)$$

با گزینش $q(h) = p(h;d,\theta)$ نابرابری بالا تنگ می‌شود. این به معنای آن است که نابرابری به برابری تبدیل می‌شود. این گام الگوریتم همانند بیشینه کردن حدپایین درست نمایی (J) نسبت به q است. در نتیجه روش کار الگوریتم امید ریاضی-بیشینه کردن به صورت زیر است:

1. پارامتر ها را مقدار آغازین $\theta(0)$ می‌دهیم.
2. تا زمان همگرایی به بیشینه محلی ادامه می‌دهیم :

$$q^{(t)} = \arg\max_q \mathcal{J}(q, \theta^{(t)}) \quad :(\text{گام-ا(امید ریاضی)}) \quad .1$$
$$\theta^{(t+1)} = \arg\max_\theta \mathcal{J}(q^{(t)}, \theta) \quad :(\text{گام-ب(بیشینه کردن)}) \quad .2$$

3. مقادیر نهایی θ و q را باز گردان

این دیدگاه نسبت به الگوریتم امید ریاضی-بیشینه کردن متعلق به نیل و هینتون است. بدین ترتیب در هر گام الگوریتم، حد پایین درست نمایی کل داده ها افزایش می‌یابد تا آنجا که در یک بیشینه محلی همگرا شود. برای رهایی از بیشینه های محلی، این الگوریتم را معمولاً چندین بار با شرایط آغازین متفاوت اجرا می‌کنند.

۸۸-۲- الگوریتم تبدیل infix یک عبارت محاسبایی به postfix و prefix

توضیح: می‌توان به صورت مستقیم نمایش prefix و postfix یک عبارت محاسباتی را از روی نمایش infix آن عبارت با استفاده از یک پشته[۲] به دست آورد.

به دست آوردن نمایش postfix از روی نمایش infix با استفاده از stack[۸]:

ابتدا از سمت راست شروع به خواندن تک تک اجزای عبارت محاسباتی می‌کنیم، که به صورت infix است و به سمت چپ حرکت می‌کنیم.

یک استک را برای نگه داری عملگرهای در نظر می‌گیریم. اگر جزیی را که از عبارت خوانده ایم یک عدد یا متغییر است (عملوند) آن را به صورت مستقیم در خروجی چاپ می‌کنیم. اگر عملگر یا پرانتز بود کارهای زیر را انجام می‌دهیم:

اگر پرانتز بسته ")" بود آن را در استک push می‌کنیم.

[2] stack [8]

اگر پرانتز باز ")" بود تا زمانی در استک به پرانتز بسته برسیم pop می‌کنیم و در سر راه هرچه عملگر دیدیم آن را در خروجی چاپ می‌کنیم.

اگر یک عملگر بود و در سر استک پرانتز بسته ")" بود عملگر را در استک push می‌کنیم.

اگر عملگر بود و در سر استک یک عملگر دیگر موجود بود دو حال وجود دارد اگر عملگری که دیدیدم از عملگری که در سر استک است، اولویت کمتری داشت عملگر سر استک را از استک pop می‌کنیم و آن را در خروجی چاپ می‌کنیم.

دوباره چک می‌کنیم آیا عملگری که دیده‌ایم با چیزی که سر استک هست اولویتش کمتر است یا نه اگر کمتر بود

دوباره همین کار را می‌کنیم تا زمانی که اولویتش کمتر از عملگر سر استک نباشد

بعد عملگری را که دیده ایم در استک push می‌کنیم، حالت دوم زمانی اتفاق می‌افتد که عملگری را که دیده‌ایم اولویت بیشتری از عملگر سر استک برخوردار باشد در این صورت عملگر جدیدی را که دیده ایم در استک push می‌کنیم.

اگر تک تک اجزای عبارت را از سمت راست به چپ را دیدیم ولی استک خالی نشده بود تمام محتویات استک را تا زمانی که خالی شود pop می‌کنیم و عملگرها را در خروجی نمایش می‌دهیم.

۸۹-۲- **الگوریتم** DES

توضیح: در DES طول قطعات ۶۴ بیت است. کلید نیز شامل ۶۴ بیت است ولی در عمل تنها از ۵۶ بیت آن استفاده می‌شود و از ۸ بیت دیگر فقط برای چک کردن parity استفاده می‌شود. الگوریتم شامل ۱۶ مرحله مشابه است که هر مرحله یک دور۴ نامیده می‌شود. متنی که قرار است رمزگذاری شود ابتدا در معرض یک جایگشت اولیه(IP) قرار می‌گیرد.

سپس یک سری اعمال پیچیده وابسته به کلید روی آن انجام می‌شود و در نهایت در معرض یک جایگشت نهایی (FP) قرار می‌گیرد IP, FP .معکوس هم هستند FP عملی که توسط IP انجام شده‌است را خنثی می‌کند. بنابراین از جنبه رمزنگاری اهمیت چندانی ندارند و برای تسهیل نمودن بار کردن قطعات داده در سخت‌افزارهای دهه ۱۹۷۰ استفاده شدند ولی اجرای DES در نرم‌افزار را کند کردند .

قبل از دور اصلی، داده به دو بخش ۳۲ بیتی تقسیم می‌شودکه این دو نیمه به طور متناوب مورد پردازش قرار می‌گیرند این تقاطع به عنوان شکل فیستل شناخته می‌شود. ساختار فیستل تضمین می‌کند که رمزگذاری و رمزگشایی دو رویه کاملا مشابه هم هستند و تنها تفاوت آنها این است که زیر کلیدها در زمان رمزگشایی در

جهت معکوس رمزگذاری به کار برده می‌شوند. و بقیه الگوریتم درهر دو یکسان است که این امر پیاده سازی رابه خصوص در سخت‌افزاربسیار آسان می‌کند و دیگر نیازی به الگوریتم‌های متفاوت برای رمزگذاری و رمزگشایی نیست. تابعی که خروجی IP را می‌گیرد وپس از شانزده مرحله ورودی FP را فراهم می‌کند تابع F نامیده می‌شود[1] .

این تابع یک ورودی ۳۲ بیتی و یک ورودی ۴۸ بیتی دارد و یک خروجی ۳۲ بیتی تولید می‌کند. بلاک ورودی شامل ۳۲ بیت که نیمه سمت چپ را تشکیل می‌دهد و با L نشان داده می‌شود و به دنبال آن ۳۲ بیت دیگر که نیمه راست را تشکیل می‌دهد و با R نمایش داده می‌شود است پس کل بلاک را می‌توان به صورت LR نمایش داد. [1]

اگر K یک بلاک ۴۸ بیتی باشد که از کلید اصلی ۶۴ بیتی مشتق شده‌است و خروجی یک دور با ورودی LR و خروجی L1R1 به صورت زیر تعریف می‌شود L1=R R1=L XOR F(R,K) .اگر KS تابعی باشد که کلید ۶۴ بیتی KEY و یک عدد صحیح در محدوده ۱ تا ۱۶ را به عنوان ورود ی می‌گیرد و کلید ۴۸ بیتی Kn را به عنوان خروجی تولید می‌کند به طوری که بیتهای Kn از تغییر محل بیتهای KEY حاصل شده‌اند داریم:

```
Kn=KS(n.KEY)
```

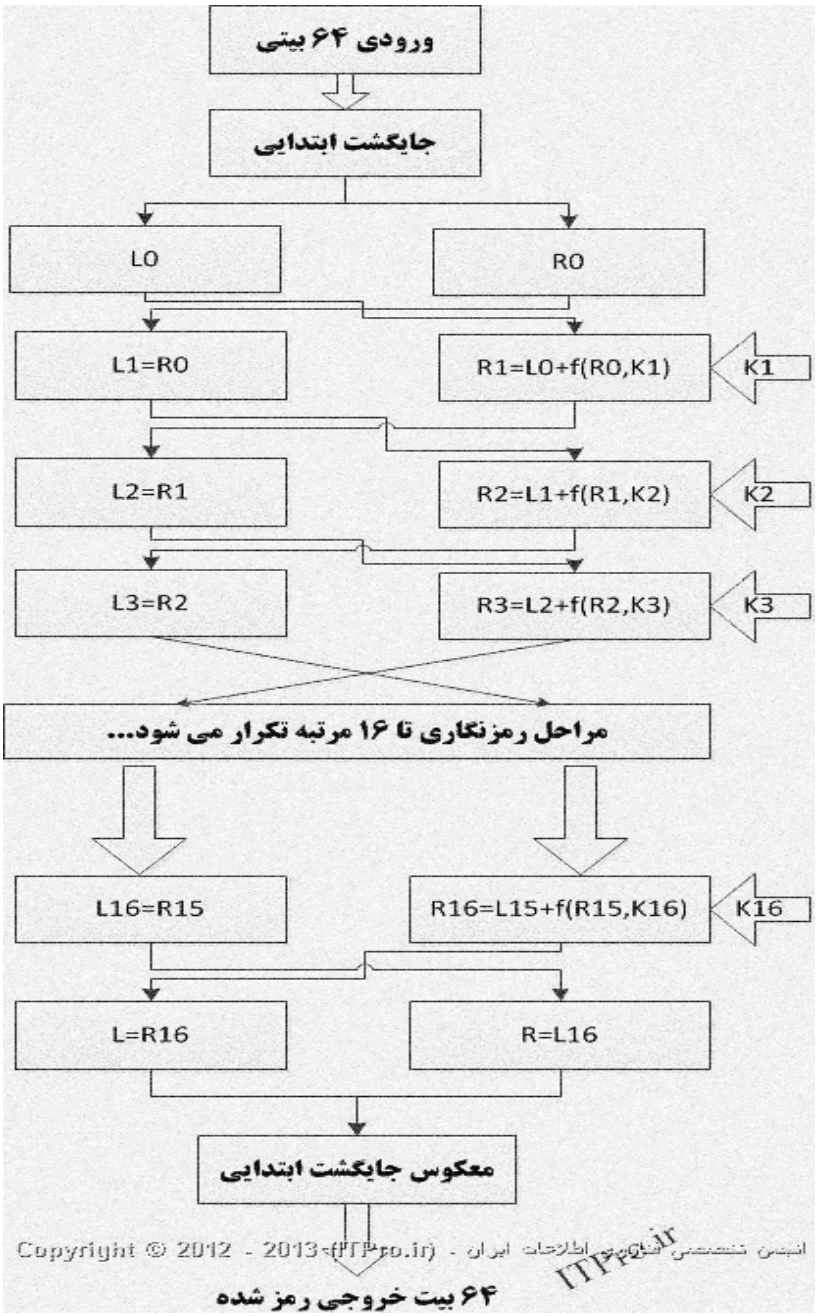
KSرا تابع key schedule می‌نامند. بنابراین در حالت کلی داریم Ln=Rn-1 Rn=Ln-1 : XOR f(Rn-1,Kn) برای رمزگشایی نیز داریم:

```
R=L1 L=R1 XOR f(L1,K)
```

در نتیجه رمزگشایی با همان الگوریتمی‌که برای رمزگذاری استفاده شد انجام می‌شودو در هر مرحله همان K بیتی که به عنوان کلید برای رمزگذاری استفاده شده بود مورد استفاده قرار می‌گیرد بنابراین می‌توان نوشت:

```
Rn-1=Ln Ln-1=Rn XOR f(Ln,Kn)
```

برای محاسبات رمزگشایی R16L16 ورودی IP و R0L0 ورودی FP است. کلید شانزدهم در مرحله اول، کلید پانزدهم در مرحله دوم و به همین ترتیب کلید اول در مرحله شانزدهم مورد استفاده قرار می‌گیرد. [3]



۹۰-۲- الگوریتم پروتکلTSL

توضیح: پروتکل TLS به برنامه‌های Client/Server اجازه می‌دهد که در شبکه از طریقی که از شنود[۲۹]جعل پیام[۳۰]جلوگیری می‌کند با یکدیگر ارتباط برقرار کنند. احراز هویت[۳۱]و ارتباط مطمئن[۳۲]در اینترنت را از طریق استفاده از رمز نگاری[۳۳]فراهم می‌کند.

Client باید برای Server مشخص کند که آیا می‌خواهد یک اتصال TLS داشته باشد با نه. دو راه برای رسیدن به این هدف وجود دارد: یک راه این است که از شماره پورت متفاوتی برای اتصال TLS استفاده شود (برای مثال پورت ۴۴۳ پروتکل امن انتقال ابرمتن) و دیگر اینکه اختصاص یک پورت مشخص از طریق سرور به کلاینت، که کلاینت آن را درخواست کرده باشد با استفاده از یک مکانیسم پروتکل خاص (برای مثال STARTTLS).

زمانی که کلاینت و سرور تصمیم گرفتند از اتصال TLS استفاده کنند، به مذاکره با استفاده از روش handshaking می‌پردازند. سپس سرور و کلاینت بر روی پارامترهای مختلفی که برای ایجاد امنیت اتصال استفاده می‌شود به توافق می‌رسند:

1) کلاینت اطلاعاتی را که سرور برای برقراری ارتباط با استفاده از SSL به آن نیاز دارد را ارسال می‌کند. مانند: شماره نسخه SSL کلاینت، تنظیمات رمزگذاری و سایر اطلاعاتی که سرور ممکن است به آن نیاز داشته باشد. [3]

۲) سرور اطلاعاتی را که کلاینت برای برقراری ارتباط با استفاده از SSL به ان نیاز دارد را برایش ارسال می‌کند. مانند: شماره نسخه SSL سرور، تنظیمات رمزگذاری و سایر اطلاعاتی که کلاینت به آن نیاز دارد. سرور همچنین گواهینامه خود را برای کلاینت ارسال می‌کند و اگر کلاینت درخواست منبعی از سرور داشته باشد، کلاینت باید احراز هویت شود و باید گواهینامه کلاینت برای سرور ارسال شود.

3) با اطلاعات دریافتی از سرور، کلاینت می‌تواند سرور را احرازهویت کند. اگر سرور تصدیق نشود، به کاربر هشدار داده می‌شود که عمل رمزگذاری و تصدیق نمی‌تواند انجام گیرد. اگر سرور به درستی تصدیق شد کلاینت به مرحله بعد می‌رود.

---

[2] eavesdropping [9]
[3] message forgery [0]
[3] authentication TLS [1]
[3] communications confidentiality [2]
[3] cryptography [3]

4) با استفاده از اطلاعات به دست آمده، کلاینت یک pre-master secret ایجاد کرده و آن را ب سرور ارسال می‌کند.

5) اگر سرور از کلاینت بخواهد هویتش را ثابت کند، کلاینت کلیه اطلاعات لازم و گواهی خود را برای سرور ارسال می‌کند.

6) اگر کلاینت تصدیق نشود، ارتباط قطع می‌شود اما اگر به درستی تصدیق شود، سرور از کلید خصوصی خود برای باز کردن pre-master secret استفاده می‌کند.

7) کلاینت و سرور از master secret برای تولید کلید جلسات استفاده می‌کنندکه یک کلید متقارن است و برای رمزگذاری و رمزگشایی اطلاعات مبادله شده استفاده می‌شود.

8) وقتی کلاینت پیغامی برای سرور ارسال می‌کند با استفاده از کلید جلسه آن را رمز می‌کند.

9) وقتی سرور پیغامی برای کلاینت ارسال می‌کند با استفاده از کلید جلسه آن را رمز می‌کند.

اکنون SSL handshake کامل است و ارتباط شروع می‌شود. کلاینت و سرور از کلید جلسه برای رمزگذاری و رمزگشایی اطلاعاتی که برای هم می‌فرستند استفاده می‌کنند.

اگر یکی از قدم‌های بالا با شکست مواجه شود TLS دچار شکست شده و ارتباط برقرار نمی‌شود.

در قدم سوم مشتری باید گواهی سرور را به درستی چک کند تا باعث بروز مشکل نشود.

۹۱-۲- الگوریتمHTTP

توضیح: وقتی شما وارد یک سایت مثل gmail.com که از پروتکل https استفاده می‌کند، می‌شوید، مرورگر شما ابتدا یک سر به سرور آن سایت می‌زند و یک گاوصندوق در-باز از سرور درخواست می‌کند: سرور جان! صاحبِ من می‌خواهد یک قطعه طلا به تو بفرستد، لطفاً یک گاوصندوقِ در-باز بفرست... منظور از گاوصندوقِ در-باز این است: سرور، با استفاده از الگوریتم RSA روال زیر را طی می‌کند:

۱. دو عدد اول بزرگ p و q را به صورت تصادفی بیابید به طوری که $p \neq q$.
۲. عدد n را محاسبه کنید به طوری که $n = pq$.
۳. تابع فی را محاسبه کنید به طوری که $\varphi(n) = (p-1)(q-1)$
۴. عدد e را انتخاب کنید به طوری که $1 < e < \varphi(n)$ و نسبت به $\varphi(n)$ اول باشد.
- عدد e به عنوان توان کلید عمومی منتشر می‌شود.

۵. عدد d را طوری بیابید که $de \equiv 1 \pmod{\varphi(n)}$ (باقی‌مانده ضرب دو عدد d و e نسبت به $\varphi(n)$ برابر ۱ باشد، به صورت: $de = 1 + k\varphi(n)$ به ازای kهای طبیعی)
- عدد d به عنوان توان کلید خصوصی محافظت می‌شود.

- دو عدد اول می‌توانند توسط روش **پیدا کردن اعداد اول احتمالی** پیدا شوند.
- معمولاً عدد عمومی (e) را در حدود 2^{16} انتخاب می‌کنند. البته بعضی از برنامه‌ها اعداد کوچکی را انتخاب می‌کنند که باعث سریعتر شدن و البته خطرات امنیتی در رمزنگاری می‌شود.
- **کلید عمومی** تشکیل می‌شود از:
  - عدد n (عدد مشترک)
  - عدد e (عدد عمومی)
- **کلید خصوصی** تشکیل می‌شود از:
  - عدد n (عدد مشترک)
  - عدد d (عدد خصوصی)
- کلید خصوصی به صورت‌های دیگری غیر از d ممکن است نگهداری شود.
  - p و q: اعداد اول برای ساختن کلید.
  - $d \mod (p-1)$ و $d \mod (q-1)$.
  - $q^{-1} \mod (p)$.
- در تمام مراحل باید اجرای کلید خصوصی سری نگه داشته شود، دو عدد p و q اگر به عنوان صورتی از کلید خصوصی نگهداری نشود بهتر است به شیوه‌ای امن نابود شوند. زیرا با این دو عدد تمام اعداد n و e، d قابل محاسبه خواهند بود.

و نهایتاً خروجی آن روال، دو کلید می‌شود: کلید عمومی[۳۴]و کلید خصوصی[۳۵]کلید عمومی‌همان گاوصندوقِ در باز است که به مرورگر شما فرستاده می‌شود و اگر در راه، دست راهزن بیفتد هیچ کاربردی ندارد (و چه بسا اگر از آن استفاده کند، خودش را بیچاره کرده! چون داده‌هایش را در صندوقی گذاشته که کلیدش را ندارد) و کلید خصوصی همان کلید گاوصندوق است که پیش سرور می‌ماند.

مرورگر شما داده‌ها را با کلید عمومی‌رمزنگاری می‌کند و به سرور می‌فرستد. اگر در راه، داده‌های شما دست راهزن بیفتد، هیچ جای نگرانی نیست! چون یک مشت چرت‌وپرت را خواهد دید و تا وقتی کلید خصوصی وجود نداشته باشد، این چرت‌وپرت‌ها به داده‌ی قابل فهم تبدیل نمی‌شوند.

(دقت کنید که منظور از راهزن، می‌تواند یک نفر مثل شخصی باشد که در کافی‌نت به عنوان مسؤول نشسته است یا شاید یک جوان در ISP شما یا شاید حتی یک نفر در کمیته فیلترینگ و یا یک نفر در مرکز داده‌ی کشورهای همسایه که داده‌های شما از آنجا عبور می‌کند تا به سایت مقصد برسد یا حتی یک نفر در شرکت گوگل!؟ هر کسی که داده‌های شما از سیستم او عبور می‌کند می‌تواند راهزن به حساب آید)

وقتی داده‌های رمزنگاری شده به سرور سایت رسید، سرور با کلید خصوصی که نزد خودش دارد آن‌ها را رمزگشایی می‌کند و مثلاً نام کاربری و رمز عبور شما را از آن استخراج می‌کند و بررسی می‌کند که صحیح است یا خیر و اگر صحیح بود، حالا می‌خواهد ایمیل‌های شما را به سیستم شما بفرستد... مجدداً این روال اما برعکسِ

[3] Public Key
[3] Private Key
[4]
[5]

آن اتفاق می‌افتد. یعنی سرور به مرورگر شما می‌آید و درخواست یک گاوصندوق در باز می‌کند و مرورگر شما با الگوریتم RSA کلیدهای عمومی‌و خصوصی را تولید می‌کند و کلید عمومی‌را به سرور می‌فرستد.

۹۲-۲- الگوریتمRSA

توضیح: آراس‌ای به طور کلی از دو کلید تشکیل می‌شود. کلید عمومی‌و کلید خصوصی. کلید عددی ثابت است که در محاسبات رمزنگاری استفاده می‌شود. کلید عمومی‌برای همه معلوم بوده و برای رمز کردن پیام استفاده می‌شود. این پیام فقط توسط کلید خصوصی باز می‌شود. به عبارتی دیگر همه می‌توانند یک پیام را رمز کنند اما فقط صاحب کلید خصوصی می‌تواند پیام را باز کند و بخواند.

مراحل زیر برای تولید کلید طی می‌شود:

1) دو عدد اول بزرگ p و q را به صورت تصادفی بیابید به طوری که $p \neq q$

2) عدد n را محاسبه کنید به طوری که $n = p\ q$

3) تابع فی را محاسبه کنید به طوری که $\varphi(n) = (p-1)(q-1)$

4) عدد e را انتخاب کنید به طوری که $1 < e < \varphi(n)$ و نسبت به $\varphi(n)$ اول باشد.

عدد e به عنوان توان کلید عمومی‌منتشر می‌شود.

عدد d را طوری بیابید که $d\ e \equiv 1 \pmod{\varphi(n)}$ باقی‌مانده ضرب دو عدد d و e نسبت به $\varphi(n)$ برابر ۱ باشد، به صورت $d\ e = 1 + k\varphi(n)$: به ازای‌kهای طبیعی)

عدد d به عنوان توان کلید خصوصی محافظت می‌شود.

دو عدد اول می‌توانند توسط روش پیدا کردن اعداد اول احتمالی پیدا شوند.

معمولاً عدد عمومی (e) را در حدود ۲۱۶ انتخاب می‌کنند. البته بعضی از برنامه‌ها اعداد کوچکی را انتخاب می‌کنند که باعث سریع‌تر شدن و البته خطرات امنیتی در رمزنگاری می‌شود.

کلید عمومی‌تشکیل می‌شود از :

- عدد n (عدد مشترک)
- عدد e (عدد عمومی)

کلید خصوصی تشکیل می‌شود از :

- عدد n (عدد مشترک)
- عدد d (عدد خصوصی)

کلید خصوصی به صورت‌های دیگری غیر از d ممکن است نگهداری شود.

- p و q: اعداد اول برای ساختن کلید.
- $d \mod (p-1)$ و $d \mod (q-1)$، d
- $q^{-1} \mod (p)$.

در تمام مراحل باید اجزای کلید خصوصی سری نگه داشته شود، دو عدد p و q اگر به عنوان صورتی از کلید خصوصی نگهداری نشود بهتر است به شیوه‌ای امن نابود شوند. زیرا با این دو عدد تمام اعداد n و e ، d قابل محاسبه خواهند بود.

1. انتخاب دو عدد اول مانند:
$q = 53$ and $p = 61$
2. محاسبه $n = pq$:
$n = 61 \times 53 = 3233$
3. محاسبه تابع فی با تولید $\varphi(n) = (p-1)(q-1)$ خواهدشد:
$\varphi(3233) = (61-1)(53-1) = 3120$
4. انتخاب هر عددی $1 < e < 3120$ که نسبت به ۳۱۲۰ اول باشد.
در نظر مي گيريم $e = 17$
5. محاسبه d, the وارون ضربی (هم‌نهشتی) $e \pmod{\varphi(n)}$,
$d = 2753$
$e \times d \mod \varphi(n) = 1$
$17 \times 2753 \mod 3120 = 1$

کلید عمومی ($n = 3233$, $e = 17$) برای پیام m هست. بنابران تابع رمز به صورت زیر است:

$$c(m) = m^{17} \mod 3233$$

کلید خصوصی ($d = 2753$) هست. برای متن رمز c تابع رمزگشایی به صورت زیر خواهد بود:

$$m(c) = c^{2753} \mod 3233$$

برای نمونه در رمز گشایی $m = 65$, را حساب می‌کنیم.

$$c = 65^{17} \mod 3233 = 2790$$

برای رمزگشایی $c = 2790$, را حساب می‌کنیم.

مثال :

درRSA ، به جفت عدد (e,n) که متن به کمک آن رمز می‌شود، اصطلاحا کلید عمومی[۳۶] به جفت عدد (d,n) که متن بوسیله آن از رمز در می‌آید، کلید خصوصی[۳۷] گفته می‌شود. نکته اساسی در RSA آن است که جهت تضمین وارون پذیری روش رمز، اعداد و بایستی در رابطه e.d mod n=x (x) صدق کنند لذا باید در انتخاب اعداد دقت کرد.

اصل اساسی دیگری که باید در رمزنگاری RSA حتما رعایت شود آن است که کدهای Pi که به هر بلوک نسبت می‌دهیم باید در شرط ۰ ≤ Pi< n صدق کند. بنابر این اگر بلوک‌ها بصورت رشته‌های k بیتی مدل شوند، باید شرط K< n ۲ برقرار باشد. دلیل این امر آن است که براحتی بتوان گزاره n Pi mod Pi=را نوشت و الا در حالت کلی این گزاره درست نمی‌باشد و در این صورت رمزگشایی صحیح داده‌ها تضمین نخواهد شد.

روش انتخاب e و dکه توسط ابداع کنندگان RSA پیشنهاد شده، عبارت است از:

الف)دو عدد دلخواه (اما بزرگ p و q) را انتخاب می‌شود.

ب)اعداد n و z را طبق دو رابطه زیر محاسبه می‌گردد:

- n = p * q
- z = (p-۱) * (q-۱)

ج) عدد d طوری انتخاب می‌شود که نسبت به z اول باشد یعنی هیچ عامل مشترکی که هر دو بر آن بخشپذیر باشند یافت نشود.

د)بر اساس d، عدد e طوری انتخاب می‌شود که رابطه زیر برقرار باشد: (بعبارتی معکوس ضربی d در پیمانه zمحاسبه شده و e نامیده می‌شود)

(e * d) mod z = ۱

آنچه که مشخص است در کاربردهای عملی، اعدادpوq حداقل صد رقمی (صد رقم در مبنای ده) انتخاب می‌شوند یعنی این دو عدد حداقل از مرتبه ۱۰۱۰۰ هستند. در این حالت عدد صحیح متناظر با بلوک‌های Pi که طبق شرط فوق باید کمتر از n باشند، نبایستی از ۸۳ کاراکتر بیشتر باشند، زیرا:

[3] public key [6]
[3] private key [7]

p , q ≈ ۱۰۱۰۰ → n = p * q ≈ ۱۰۲۰۰ → (Pi <(۲۶۶٤≈۱۰۲۰۰)) → Pi <۲۶۶٤

پس هر بلوک متن بایستی حداکثر۶۶۴ بیت یا ۸۳ کاراکتر هشت بیتی باشد. در اینجا توجه به این نکته ظریف لازم است که برای محاسبه Ae mod n لازم نیست که A به تعداد e بار در خودش ضرب و سپس باقیماده اش بر n پیدا شود زیرا با استفاده از برخی خواص ریاضی نتیجه محاسبات هیچگاه از n فراتر نمی‌رود. حال فرض کنید یک نفذگر بخواهد با در اختیار داشتن کلید عمومی(e , n)، را بدست آورد.در اینصورت باید در وهله اول nرا به دو عامل اول آن یعنی p و q تجزیه کند تا بتواند z را محاسبه کرده و سپس d را بدست آورد. برای تجزیه اعداد به عوامل اول آن هیچ راهی بجز جستجو و آزمون وجود ندارد و با توجه به این که n حداقل دویست رقمی‌است،تجزیه چنین اعدادی حتی به کمک کامپیوتر هزاران سال طول خواهد کشید.

اگر چه تحقیق بر روی مسئله تجزیه اعداد بزرگ به عوامل آن هنوز ادامه دارد اما هنوز الگوریتم کارآمدی که بتواند اعداد بزرگ را با هر طولی در زمان ثابت یا در حد متعارف کوچکی به عوامل اول آن تجزیه کند، یافت نشده است، لذا با گذشت ۳۰ سال از معرفی RSA هنوز از شان آن کاسته نشده است بلکه فقط کلیدها به جهت محکم کاری بزرگتر شده اند .

از آنجا که اعداداول هیچ نظم شناخته شده ای ندارند لذا انتخاب اعداد اول بسیار بزرگ p و q یکی از چالش‌های بزرگ RSA است زیرا برای اثبات اول بودن عددی مثل p باید محدوده اعدادکمتر یا مساوی $\sqrt{p}$ بررسی و بخش ناپذیری p بر آنها مطالعه گردد که هرچه p بزرگتر باشد محدوده جستجوی $\sqrt{p}$ بزرگتر خواهد بود. برای مثال محدوده جستجوی عددی ۵۱۲ بیتی از مرتبه ۲۲۵۶ می‌شود که جستجوی چنین فضایی عملا غیر ممکن است. بنابر این تنها راه چاره استفاده از یکسری از قضایای ریاضی است که به ما کمک می‌کنند محدوده جستجو را کوچکتر نموده و مراحل حدس زدن کمتر شود. [4]

الگوریتم تولید اعداد تصادفی

توضیح: اگر دنباله مقادیر پذیرفته شده توسط هر ویژگی از توزیع احتمال معینی پیروی کند، آن دنباله مقادیر تصادفی است. تصمیمات مبتنی بر مقادیر موجود در دنباله تصادفی شمرده می‌شود و در نتیجه شبیه‌سازی برخوردار از رفتار تصادفی خواهد بود.

خواص اعداد تصادفی

هر دنباله از اعداد تصادفی مانند.... ,R1 ,R2 باید دو خاصیت آماری مهم داشته باشدکه عبارتند از تابع تجمعی، امید ریاضی و واریانس Ri به ترتیب عبارتند از:

پیامدهای خاصیت استقلال و خاصیت توزیع احتمال یکنواخت پیوسته برای اعداد تصادفی:

1) اگر فاصله (۱و۰) به n رده یا زیر فاصله مساوی تقسیم شود،انتظار می‌رود که از N مشاهده آنها در هر رده قرار گیرد.

2) احتمال حصول یک مشاهده در یک رده خاص مستقل از محل قرار گرفتن سایر پیشامدهاست.

تولید اعداد شبه تصادفی

مقصود از شبه تاکید بر این مطلب است که استفاده از یک روش قطعی و مشخص برای اعداد تصادفی، امکان بالقوه تصادفی بودن واقعی را از بین می‌برد. پس اعداد تولید شده واقعاً تصادفی نیست.هدف هر روش اینست که به نحوی اعداد تصادفی در محدوده (۱و۰) تولید کند تا دو خاصیت استقلال و توزیع یکنواخت را داشته باشد.

نمونه‌های خطاهایی که بدلیل خواص استقلال و توزیع احتمال یکنواخت به هنگام تولید اعداد شبه تصادفی ایجاد می شوند عبارتند از:

1) اعداد تصادفی ممکن است توزیع احتمال یکنواخت نداشته باشد.

2) اعداد تصادفی ممکن است جدا از هم باشد.

3) میانگین اعداد تولید شده ممکن است بیش از حد بزرگ یا بیش از حد کوچک باشد.

4) واریانس اعداد تصادفی تولید شده ممکن است تفاوت قابل توجهی از مقدار متعارف داشته باشد.

5) ممکن است دنباله اعداد تولید شده تغییراتی تناوبی از خود نشان دهد.

مانند:

الف)وجود همبستگی بین اعداد.

ب)وجود رابطه مقداری بین اعداد مجاور به این ترتیب که اعداد مجاور روندی صعودی یا نزولی از خود نشان دهد.

ج)وجود چند عدد بزرگتر از میانگین و به دنبال آن وجود چند عدد کوچکتر از میانگین.

در بررسی‌های شبیه سازی، اعداد تصادفی بوسیله یک کامپیوتر رقمی تولید

می شود.

1) ویژگی‌های الگوریتم‌های تولید اعداد تصادفی:

2) روش یا الگوریتم تولید اعداد تصادفی باید سریع باشد.

3) الگوریتم نباید نیاز به مقدار زیادی حافظه کامپیوتری داشته باشد.

4) طول دنباله اعداد تولید شده باید به اندازه کافی بلند باشد.

5) اعداد تصادفی تولید شده توسط یک الگوریتم باید در صورت نیاز تکرار پذیر باشد.

6) اعداد تصادفی تولید شده باید تا حدود زیادی از خواص آماری توزیع یکنواخت و استقلال برخوردار باشد.

۹۳-۲- روش‌های مختلف تولید اعداد تصادفی

روش میان مربعی

این روش با یک عدد اولیه به نام هسته شروع بکار می کند. روال کار چنین است که هسته را مربع می کنند و ارقام میانی آن را تعیین و پس از نوشتن صفر-ممیز در سمت چپ ارقام میانی، اولین عدد تصادفی را تولید می‌کنند. به منظور تولید عدد تصادفی دوم باید ارقام میانی در مرحله قبل را مربع، ارقام میانی حاصل را تعیین وبه اعداد اعشاری تبدیل کردو... اگر هسته n رقمی باشد، مربع آنرا ۲ n تا ۲ n-1 رقمی خواهد بود و اگر n زوج باشد، باحذف ارقام از هر سمت مربع آن می توان ارقام میانی را تعیین کرد. بزرگترین عدد n رقمی، بزرگترین طول دنباله را مشخص می‌کند.

عمده مسائل ایجاد شده از روش میان مربعی

مثال 7-1 هسته= X_0و مقدار 5497 iامین عددي كه مربع مي شود X_i و iامین عدد تصادفي كه توليد مي شود R_i باشد.

$X_0=5497$, $X_0^2=(5497)^2=30217009$ $X_1=2170$ $R_1=0.2170$

$X_1^2=(2170)^2=04708900$ $X_2=7089$ $R_2=0.7089$

$X_2^2=(7089)^2=50253921$ $X_3=2539$ $R_3=0.2539$

1) نمی توان قوائد ساده ای برای تعیین مقدار هسته ارائه کرد.

2) با ظهور رقم صفر در سمت چپ ارقام میانی، دنباله اعداد تصادفی تولید شده به سرعت به انتها

3) می‌رسد.

4) از هم پاشیده شدن الگوریتم بدلیل حصول مقدار تکراری یا مقدار صفر برای اعداد میانی

در این روش با انتخاب دو هسته X0 و X0′ که تعداد ارقامشان مساوی است شروع بکار می‌کند.

اگر دو هسته n رقمی باشد،آنها را در هم ضرب می‌کنیم و با حذف از ارقام سمت راست و همین تعداد رقم از سمت چپ حاصلضرب، nرقم قرار گرفته در وسط را تعیین و آنرا تبدیل به یک عدد اعشاری کوچکتر از یک می‌کنیم. اگر n قرار گرفته در وسط حاصلضرب باX0 و X1 تکرار می‌کنیم و...

مثال 4-7

$X_0=7229 , X_0'=2938$

$U_1=X_0'X_0=(2938)(7229)=21238802\ X_1=2388\ R_1=0.2388$

$U_2=X_0X_1=(7229)(2388)=17262852\ X_2=2628\ R_2=0.2628$

$U_3=X_1X_2=(2388)(2628)=6275664\ X_3=2756\ R_3=0.2756$

۹۴-۲- الگوریتم جی اس ام – آی ۱/ ۵(GSM)

توضیح: ۱/۵A یک رمز دنباله‌ای است که در جهت حفظ حریم خصوصی در تلفن‌های همراه استاندارد جی‌اس‌ام به کار رفته است. این سیستم رمزنگاری در ابتدا به شکل محرمانه ارائه شد، امّا بعداً به‌واسطه مهندسی معکوس به شکل عمومی‌شناخته شد. ضعف‌های جدی و قابل توجّه‌ای در این سیستم رمزنگاری وجود دارد.

در سیستمA ۱/۵ در هر ۴،۶۱۵ هزارم ثانیه، ۱۱۴ بیت ارسال می‌شود. این ۱۱۴ بیت با دنباله ۱۱۴ بیتی که از یک مولد خارج می‌شود، XOR شده و سپس مدوله شده و ارسال می‌شود A . ۱/۵ برای فعال‌سازی به یک کلید ۶۴ بیتی و یک عدد ۲۲ بیتی عمومی‌به نام شماره فریم نیاز دارد. پیاده‌سازی‌های قدیمی‌تر جی‌اس‌ام، برای تولید کلید ازComp128v ۱ استفاده می‌کردند که ده بیت کلید خروجی آن همواره برابر صفر بود و در واقع طول مؤثّر کلید در آن ۵۴ بیت بود. این ضعف با معرّفیComp128v۲ بر طرف شد. هنگام کارکردن در مد GPRS/EDGE، پهنای باند بیشتر اجازه استفاده از فریم‌های ۳۴۸ بیتی را نیز فراهم می‌کند. در این شرایط در رمز دنباله‌ای ازA ۳/۵ استفاده می‌شود. درA ۱/۵ از سه ثبات تغییر بازخورد خطی با کلاک نامنظّم استفاده می‌شود که XOR خروجی آنها، به عنوان خروجی مولد مورد استفاده قرار می‌گیرد. [1]

شماره ثبات	تعداد بیت‌ها	بیت کلاک	بیت‌های تپ شده
۱	۱۹	۸	۱۳, ۱۶, ۱۷, ۱۸
۲	۲۲	۱۰	۲۰, ۲۱
۳	۲۳	۱۰	۷, ۲۰, ۲۱, ۲۲

اعمال کلاک به هر ثبات در هر کلاک توسط قاعده اکثریت تعیین می‌شود. هر رجیستر یک بیت دارد که شیفت خوردن آن در هر کلاک از روی آن معلوم می‌شود. (این بیت با رنگ زرد در شکل نمایش داده شده است.) اگر بیت متناظر با هر ثبات با بیتی که اکثریت را بین سه بیت مذکور تشکیل می‌دهد برابر باشد،

آنگاه آن ثبات شیفت می‌خورد. بنابرین در هر سیکل، حداقل دو ثبات شیفت خورده و همچنین در هر کلاک هر ثبات به احتمال ۴/۳ شیفت می‌خورد. در ابتدا تمام ثبات‌ها با مقدار اولیّه صفر مقداردهی می‌شوند، سپس طی ۶۴ کلاک، پیش از کلاک i ام، کم‌ارزش‌ترین بیت هر رجیستر با بیت i ام کلید XOR می‌شود.

$$R[0] = R[0] \oplus K[i].$$

به طریق مشابه ۲۲ بیت بعدی هم طی ۲۲ کلاک اضافه می‌شود. سپس ثبات‌ها برای ۱۰۰ کلاک کار می‌کنند. از آن به بعد خروجی آنها برای رمز دنباله‌ای استفاده می‌شود.

۹۵-۲- الگوریتم مسیریابی فلوید وارشال[۳۸]

توضیح: الگوریتم فلوید-وارشال[۳۹] یک الگوریتم مبتنی بر روش برنامه‌نویسی پویا برای محاسبه‌ی کوتاهترین مسیر بین هر دو جفت گره گراف‌های وزن‌دار است. دو الگوریتم رایج دایکسترا و بلمن-فورد روش‌های محاسبه‌ی کوتاهترین مسیر از مبدأ ثابت هستند که در صورت تکرار آنها به ازای هر گره عملکردی همانند الگوریتم فلوید-وارشال دارند. اما این الگوریتم ویژگی‌هایی دارد که آن را برجسته می‌کند:

1) پیاده‌سازی ساده‌تری دارد.
2) بر خلاف الگوریتم دایکسترا برای گراف‌هایی شامل یال با وزن منفی نیز قابل استفاده است.
3) در حالت کلی مرتبه‌ی زمانی بهتری نسبت به الگوریتم بلمن-فورد دارد.
4) همانند الگوریتم بلمن-فورد قابلیت تشخیص دور منفی را دارد.

فرض کنید ماتریس مجاورت گراف را با Mn×n و گره‌های گراف را به صورت {v 1 ,v 2 ,v 3 ,⋯,v n} نمایش دهیم. اگر ماتریس Dn×n(k) معرف ماتریس طول کوتاه‌ترین مسیر بین گره‌های گراف تنها با اجازه‌ی عبور از گره‌های {v 1 ,v 2 ,v 3 ,⋯,v k} باشد، ماتریس D(n) جواب مسأله خواهد بود. همچین D(0) همان M است (چرا؟). پس محاسبه‌ی کوتاه‌ترین مسیر بین گره‌ها به محاسبه‌ی ماتریس D(n) تبدیل می‌شود.

[8] Floyd-Warshall
[9] Floyd-Warshall

کوتاهترین مسیر از گره vi به گره vj با اجازه‌ی عبور از گره v 1 یا مسیر مستقیم بین آن دو (با اندازه‌ی M ij) است یا مسیری که از گره v1 (با اندازه‌ی Mi1+M1j) می‌گذرد؛ یعنی:

D (1) ij =min{M ij ,M i1 +M 1j }

با استفاده از این رابطه درایه‌های ماتریس D(1) از روی ماتریس مجاورت (یا همان ماتریس D(0)) قابل محاسبه است. درایه‌های ماتریس D(2) طول کوتاهترین مسیرها با اجازه‌ی عبور از گره‌های v1 و v2 است. چنین مسیرهایی یا بدون استفاده از گره v2 هستند (یعنی همان Dij (1)) یا با عبور از این گره.

اگر از این گره استفاده شود، طول کوتاه‌ترین مسیر D(1) i2+D(1) 2j خواهد بود؛ چرا که D(1) i2 طول کوتاه‌ترین مسیر از گره vi به این گره و D(1) 2j طول کوتاه‌ترین مسیر از آن به گره vj است؛ پس:

$$D_{ij}^{(2)} = min\{D_{ij}^{(1)}, D_{i2}^{(1)} + D_{2j}^{(1)}\}$$

این استدلال برای محاسبه‌ی هر کدام از ماتریس‌های D(k) برقرار است و در حالت کلی می‌توان نوشت:

$$D_{ij}^{(k)} = min\{D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)}\}\,; \qquad 0 < k \leq n\,,\ D^{(0)} = M$$

رابطه‌ی فوق مبنای کار الگوریتم فلوید-وارشال برای محاسبه‌ی طول کوتاه‌ترین مسیر بین گره‌های یک گراف است.

الگوریتم وارشال همه مسیرهای ممکن در یک گراف، بین هر جفت از راس‌ها را مقایسه می‌کند. این الگوریتم قادر است این کار را تنها با مقایسه انجام دهد. این ملاحظه قابل توجهی می‌باشد که در یک گراف یال وجود داشته باشد و هر ترکیبی از یال‌ها چک شده باشد.

یک گراف با راس‌های که i از ۱ تا N می‌باشد را در نظر بگیرید. علاوه بر این یک تابع به نام shortestPath(i,j) ، kرا در نظر بگیرید که کوتاه‌ترین مسیر ممکن از i تا j را با استفاده از راس‌های ۱ تا k که به عنوان راس‌های میانی در امتداد مسیر می‌باشند را بر می‌گرداند.

هم اکنون این تابع داده شده‌است.هدف ما پیدا کردن کوتاه‌ترین مسیر از هر i تا هر j تنها با استفاده از راس‌های ۱ تا ۱+k می‌باشد. دو کاندیدا برای این مسیر وجود دارد:

-1کوتاهترین مسیری که فقط از راس‌های موجود در مجموعه ی (1 , ... ,k)استفاده می‌کند.

-2تعدادی مسیر که از i تا k+1و سپس از k+1تا j می‌روند وجود دارد که این مسیر بهتر می‌باشد.

ما می‌دانیم که بهترین مسیر از i تا j که فقط از راس‌های بین ۱ تا k+1 استفاده می‌کند توسط i,j (ShortestPath)، kتعریف شده‌است و واضح است که اگر یک مسیر بهتر از i تا ۱+kو از ۱+kتا j وجود داشته باشد بنابراین طول مسیربین i,j ازالحاق کوتاه‌ترین مسیر از i تا ۱+kو کوتاه‌ترین مسیر از۱ +k تا j بدست می‌آید. بنابراین تابع i,j(ShortestPath)، kرا در در فرمول بازگشتی زیر ارائه می‌دهیم:

$$\text{shortestPath}(i,j,k) = \min(\text{shortestPath}(i,j,k-1), \text{shortestPath}(i,k,k-1) + \text{shortestPath}(k,j,k-1));$$

این رابطه قلب الگوریتم وارشال می‌باشد این الگوریتم در اولین محاسبه i,j (ShortestPath)، 1را برای همه جفت‌های (i ,j) پیدا می‌کند و سپس از این برای پیدا کردن i,j (Shortestpath)، 2 برای همه جفت‌های (i ,j) استفاده می‌شود و این روال تا زمانی که k=N شود ادامه می‌یابد و ما می‌توانیم کوتاه‌ترین مسیر بین جفت‌های (i,j) رابا استفاده از راس‌های میانی پیدا کنیم.

برای پیدا کردن همه n^2 تا $W_k(i,j)$(یک است اگر بین i,j راسی بزرگتر از k نباشد در غیر این صورت صفر می‌باشد.) از $W_{k-1(i,j)}$ به $2n^2$ بیت عملیات نیاز داریم. برای اینکه ما با $W_0 = W_{\mathcal{R}}$ شروع می‌کنیم و n ماتریس صفر و یک

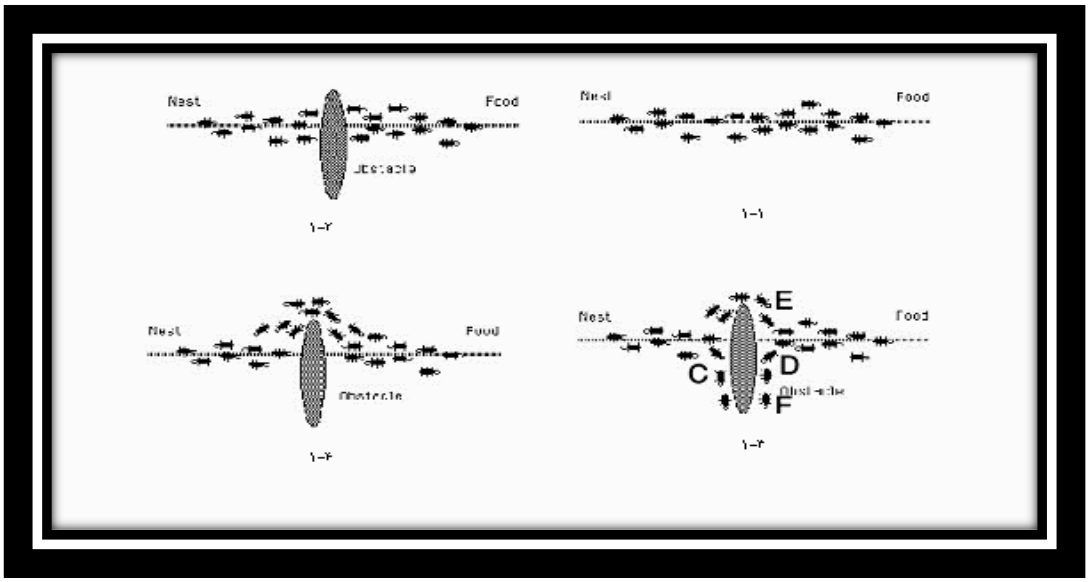
$$W_1, W_2, \ldots, W_n = M_{\mathcal{R}^*}$$

را محاسبه می‌کنیم.در نتیجه جمع بیت عملیات‌هایی که استفاده کردیم برابر با $n \times 2n^2 = 2n^3$ می‌باشد.بنابراین پیچیدگی الگوریتم از Θ(n^3) باشد.

۹۶-۲- الگوریتم کلونی مورچه‌ها

توضیح: مورچه‌ها هنگام راه رفتن از خود ردی از ماده شیمیایی فرومون[4] بجای می‌گذارند البته این ماده بزودی تبخیر می‌شد ولی در کوتاه مدت بعنوان رد مورچه بر سطح زمین باقی می‌ماند. یک رفتار پایه‌ای ساده در مورچه‌های وجود دارد :

[4] Pheromone [0]



آنها هنگام انتخاب بین دو مسیر بصورت احتمالاتی[۴۱] مسیری را انتخاب می‌کنند که فرومون بیشتری داشته باشد یا بعبارت دیگر مورچه‌های بیشتری قبلاً از آن عبور کرده باشند. حال دقت کنید که همین یک تمهید ساده چگونه منجر به پیدا کردن کوتاه‌ترین مسیر خواهد شد:

همانطور که در شکل ۱-۱ می بینیم مورچه‌های روی مسیر AB در حرکت‌اند (در دو جهت مخالف) اگر در مسیر مورچه‌ها مانعی قرار دهیم (شکل ۱-۲) مورچه‌ها دو راه برای انتخاب کردن دارند. اولین مورچه از A می‌آید و به C می‌رسد، در مسیر هیچ فرومونی نمی‌بیند بنابر این برای مسیر چپ و راست احتمال یکسان می‌دهد و بطور تصادفی و احتمالاتی مسیر CED را انتخاب می‌کند.

اولین مورچه‌ای که مورچه اول را دنبال می‌کند زودتر از مورچه اولی که از مسیر CFD رفته به مقصد می‌رسد. مورچه‌ها در حال برگشت و به مرور زمان یک اثر بیشتر فرومون را روی CED حس می‌کنند و آن‌را بطور احتمالی و تصادفی (نه حتما و قطعا) انتخاب می‌کنند. در نهایت مسیر CED بعنوان مسیر کوتاه‌تر برگزیده می‌شود.

در حقیقت چون طول مسیر CED کوتاه‌تر است زمان رفت و برگشت از آن هم کمتر می‌شود و در نتیجه مورچه‌های بیشتری نسبت به مسیر دیگر آن‌را طی خواهند کرد چون فرومون بیشتری در آن وجود دارد.

نکته بسیار با اهمیت این است که هر چند احتمال انتخاب مسیر پر فرومون توسط مورچه‌ها بیشتر است ولی این کماکان احتمال است و قطعیت نیست. یعنی اگر مسیر CED پرفرومون تر از CFD باشد به هیچ عنوان نمی‌شود نتیجه گرفت که همه مورچه‌ها از مسیر CED عبور خواهند کرد بلکه تنها می‌توان گفت که مثلا ۹۰٪ مورچه‌ها از مسیر کوتاه‌تر عبور خواهند کرد.

[4] Statistical [1]

اگر فرض کنیم که بجای این احتمال قطعیت وجود می‌داشت، یعنی هر مورچه فقط و فقط مسیر پرفرومون‌تر را انتخاب می‌کرد آنگاه اساساً این روش ممکن نبود به جواب برسد. اگر تصادفاً اولین مورچه مسیر CFD (مسیر دورتر) را انتخاب می‌کرد و ردی از فرومون بر جای می‌گذاشت آنگاه همه مورچه‌ها بدنبال او حرکت می‌کردند و هیچ وقت کوتاه‌ترین مسیر یافته نمی‌شد. بنابراین تصادف و احتمال نقش عمده‌ای در ACO بر عهده دارند.

نکته دیگر مسئله تبخیر شدن فرومون بر جای گذاشته شده است. برفرض اگر مانع در مسیر AB برداشته شود و فرومون تبخیر نشود مورچه‌ها همان مسیر قبلی را طی خواهند کرد. ولی در حقیقت این طور نیست. تبخیر شدن فرومون و احتمال به مورچه‌ها امکان پیدا کردن مسیر کوتاه‌تر جدید را می‌دهند.

«تبخیر شدن فرومون» و «احتمال-تصادف» به مورچه‌ها امکان پیدا کردن کوتاه‌ترین مسیر را می‌دهند. این دو ویژگی باعث ایجاد انعطاف در حل هرگونه مسئله بهینه‌سازی می‌شوند. مثلا در گراف شهرهای مسئله فروشنده دوره گرد، اگر یکی از یال‌ها (یا گره‌ها) حذف شود الگوریتم این توانایی را دارد تا به سرعت مسیر بهینه را با توجه به شرایط جدید پیدا کند. به این ترتیب که اگر یال (یا گره‌ای) حذف شود دیگر لازم نیست که الگوریتم از ابتدا مسئله را حل کند بلکه از جایی که مسئله حل شده تا محل حذف یال (یا گره) هنوز بهترین مسیر را داریم، از این به بعد مورچه‌ها می‌توانند پس از مدت کوتاهی مسیر بهینه (کوتاه‌ترین) را بیابند. [1]

۹۷-۲- الگوریتم بویر مور

توضیح: این الگوریتم رشته االگو را پیش پردازش می‌کند اما رشته متن را پیش پردازش نمی‌کند. این الگوریتم زمانی مناسب است که رشته الگو بسیار کوتاه‌تر از رشته متن باشد. این الگوریتم با استفاده از اطلاعاتی که از پیش پردازش الگو به دست می‌آورد بعضی از قسمت‌های متن را بررسی نمی‌کند. بنابراین نسبت به بسیاری از الگوریتم‌های دیگر ضریب ثابت پایین‌تری دارد. ایده اصلی الگوریتم این است که به جا اینکه سر الگو را با متن مقایسه کند انتهای الگو را با متن مقایسه می‌کند.

1) به کاراکتر-i ام رشته S اشاره می‌کند.

2) S[i..j] زیر رشته S از موقعیت i تا موقعیت j را نشان می‌دهد.

3) پیشوند رشته S زیر رشته S[1..i] است به طوری که i در بازه [۱]n..و n طول رشته S باشد.

4) پسوند رشته S زیر رشته S[i..n] است به طوری که i در بازه [i..n] و n طول رشته S باشد.

5) رشته‌ای را که جستجو را برای آن انجام می‌دهیم الگو می‌نامیم و با P نمایش می‌دهیم.

6) رشته‌ای را که جستجو را در آن انجام می‌دهیم متن می‌نامیم و با T نمایش می‌دهیم.

7) طول رشته P را با n نشان می‌دهیم.

8) طول رشته T را با m نشان می‌دهیم.

9) هم طرازی رشته P با T موقعیتی مانند k است به طوری که آخرین حرف P با حرف موقعیت k- ام رشته T هم طراز شود.

10) تطابق P در یک هم طرازی اتفاق می‌افتد به طوری که P با T[(k-n+1) .. k] معادل باشد. [1]

الگوریتم بویر مور به دنبال تطابق‌های رشته P در رشته T به وسیله مقایسه کاراکتر در هم طرازی‌های متفاوت می‌گردد. به جای جستجوی خام همه هم طرازی‌های متفاوت (که تعداد آن‌ها n- m+1 است)، بویر مور از اطلاعات پیش پردازش P استفاده می‌کند تا جای که امکان دارد هم طرازی‌های کمتری را بررسی کند. راه معمول جستجوی یک متن مقایسه کاراکترهای آن با اولین کاراکتر الگو است.

هنگامی‌که دو کاراکتر یکسان بودند مابقی کاراکترهای متن با کاراکترهای الگو مقایسه می‌شوند. اگر تطابقی وجود نداشت دوباره متن، کاراکتر به کاراکتر برای پیدا کردن تطابق بررسی می‌شود. بنابراین بیشتر کاراکترهای متن باید بررسی شود.

نکته کلیدی این الگوریتم این است که با مقایسه انتهای الگو با کاراکترهای متن بسیاری از کاراکترهای متن را بررسی نمی‌کند. چراکه اگر دو کاراکتر یکسان نباشند نیاز نیست تا تمام کاراکترهای دیگر را عقب گرد بررسی کنیم.

نکته جالب اینجاست که اگر این کاراکتر با هیچکدام از کاراکترهای الگو یکسان نباشد، کاراکتر بعدی که در متن نیاز به بررسی در متن دارد؛ n (طول الگو) موقعیت جلوتر است. اگر این کاراکتر در الگو وجود داشت یک انتقال جزئی صورت می‌گیرد تا این دو کاراکتر زیر هم قرار گیرند.

این انتقال‌ها باعث می‌شود که تعداد کاراکترهایی که باید بررسی شوند کاهش یابد. به طور دقیق‌تر، الگوریتم با هم طرازی k=n شروع می‌شود؛ بنابراین شروع P با شروع T هم طراز می‌شود. سپس کاراکترهای P و T از موقعیت n ام به صورت عقبگرد با هم مقایسه می‌شوند.

مقایسه ادامه می‌یابد تا هنگامی‌که یا به ابتدای p برسیم (یک تطابق پیدا کردیم)؛ یا عدم تطابق مشاهده شود که در این صورت با استفاده از قوانین زیر الگو را بیشترین مقدار ممکن انتقال می‌دهیم و مقایسه‌ها در هم طرازی جدید انجام می‌شود. این روند تا جایی ادامه می‌یابد که که الگو به مکانی بعد از انتهای T انتقال یابد که به این

معناست که تطابق جدیدی وجود ندارد. انتقال‌ها با استفاده از جداولی که از پیش پردازش P به دست می‌آید در $O(1)$ اجرا می‌شود.

۹۸-۲- الگوریتم سازگاری کمان

توضیح : سازگاری قوس یا سازگاری گنبدی نیز ترجمه شده است. الگوریتمی‌برای حل مسائل ارضای محدودیت[۴۲] می‌باشد.

در این روش سازگاری حالت‌ها با کمان نشان داده شده و سعی می‌شود با حذف تدریجی مقادیر ناسازگار جواب مناسب را پیدا کرد. اگر X و Y دو متغیر در یک مسئله ارضای محدودیت باشند آنگاه $X \rightarrow Y$ سازگار کمان است، اگر و فقط اگر به ازای تمامی‌مقادیر x ازX، مقداری مجاز برای Y موجود باشد. معروفترین الگوریتم برای این کار الگوریتم سازگاری کمان ۳ یا به اختصار AC-3 می‌باشد.

1) الگوریتم یک صف از کمان‌ها تشکیل می‌دهد.
2) هر بار یک کمان $X \rightarrow Y$ از صف انتخاب شده و سازگاری کمان برای آن بررسی می‌شود.
3) اگر دامنه X تغییری نکرد، به سراغ کمان بعدی می‌رود.
4) اگر دامنه X تغییر کرد، کمان‌های $Z \rightarrow X$ برای تمام همسایه‌های X را به صف اضافه می‌کند.
5) اگر دامنهX تهی شد، مسئله جواب ندارد.
6) الگوریتم AC-3 در بدترین مورد از مرتبه $O(ed^3)$ است که d حداکثر اعضای دامنه هر گره و c تعداد کمان‌ها است.

۹۹-۲- الگوریتم شاخه‌بندی موضوعات

برای مشاهده این الگوریتم به آدرس زیر مراجعه نمایید:

http://aftab.cc/tutorial/552

۱۰۰-۲- الگوریتم یافتن کلمات کلیدی بین اسناد

برای مشاهده این الگوریتم به آدرس زیر مراجعه نمایید:

http://aftab.cc/article/1236

[4] Constraint satisfaction problem

منابع

[1] د .و .ک .ط .الگوریتم" ,درس و کنکور طراحی الگوریتم ",مهندس حمیدرضا مقسمی ,

[2] ط .الگوریتم‌ها ,تألیف دکتر محمود نقیب‌زاده

[3] آ .س .الگوریتم‌ها" ,شیرعلی شهرضا و شیرعلی شهرضا

[4] ت .و .ط .ا ..رایانه " ,احمد فراهی، جعفر تنها ",دانشگاه پیام‌نور