طراحی الگوریتم - تحلیل الگوریتم

محسن هوشمند

فهرست

الگوریتم

به نظر واژه الگوریتم منشا از نام خوارزمی دارد. مختصری در این باره را عینا از کتاب جبر و مقابله وی نقل می‌کنیم.

فأما **الأموال والجذور التی تعدل العدد** فمثل قولک مال و عشرۀ أجذاره یعدل تسعۀ وثلاثین درهماً و معناه أی مال إذا زدت علیه مثل عشرۀ أجذاره بلغ ذلک کله تسعۀ وثلاثین.

فقیاسه أن تنصف الأجذار و هی فی هذه المسألۀ خمسۀ فتضربها فی مثلها فتکون خمسۀ وعشرین فتزیدها علی التسعۀ والثلاثین فتکون أربعۀ وستین فتأخذ جذرها و هو ثمانیۀ فتنقص منه نصف الأجذار و هو خمسۀ فیبقی ثلاثۀ و هو جذر المال الذی ترید و المال تسعۀ[1].

جهت فهم مطلب اصطلاحات جذر معادل x، مال معادل x^2 است. ترجمه سخن خوارزمی را در ادامه می‌آوریم[2].

«اما مال‌ها و جذرهایی که با عددی برابر می‌شوند:

اگر بگویی: یک مال، به اضافۀ ده جذر از آن مال، با سی و نه درهم برابر می‌شود، مقصود آن است که اگر به مالی به اندازۀ ده جذر از آن مال افزوده شود مجموع آن می‌شود سی و نه درهم.

راه حل آن چنین است: باید جذرها را نصف کنی – مقدار نصف آن در این مسئله پنج می‌شود – و آن نصف را در مانند خودش ضرب کنی، در این صورت حاصل‌ضرب بیست و پنج می‌شود. آن‌گاه این عدد را بر سی و نه بیفزایی، مجموعه شصت و چهار

[1] بخش شش از- محمد بن موسی الخوارزمی، جبر و مقابله ۲۱۵ ق، ۲۰۹ ش، ۸۳۰ م
[2] م خوارزمی، جبر مقابله، حسین خدیوجم، ۱۳۶۳

می‌شود، سپس جذر این عدد را می‌گیری، هشت می‌شود، آن‌گاه نیمی از شمارهٔ جذرها را که عبارت باشد از پنج، از آن کم می‌کنی که در نتیجه سه باقی می‌ماند، و همین عدد سه، جذر مال مورد نظر است، و آن مال نه است.

تلخیص سخن شیخ: توصیف الگوریتم برای حل نوع خاصی از معادلات درجه دو با استفاده از مثال $x^۲ + ۱۰x = ۳۹$.

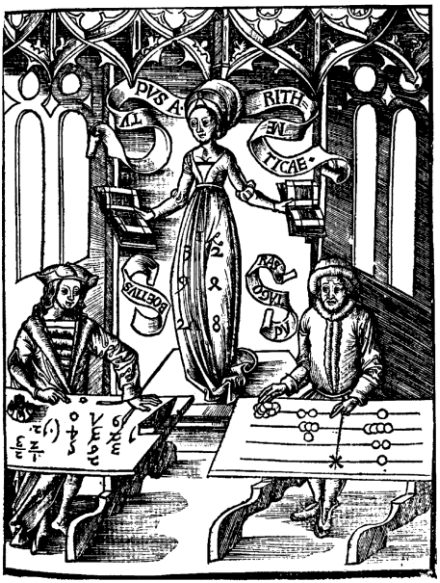
$$x^۲ + ۱۰x = ۳۹ \Longrightarrow \left(x + ۵\right)^۲ = ۳۹ + ۲۵ = ۶۴;$$

$$x + ۵ = ۸ \Longrightarrow x = ۳; x^۲ = ۹$$

روش حل بالا را می‌توان از نمونه‌های حل الگوریتمی دانست.

وجود الگوریتم‌ها از زمانی که اطلاع داریم و از زمانی پیش از اطلاق نام الگوریتم بدان‌ها وجود داشته‌اند.

روش حل خوارزمی با ترجمه به لاتین و اروپا منتقل می‌شود. منجر به تلاش افرادی با حساب عددی می‌شود. در نتیجه اندک اندک به معنای هر گونه روش محاسباتی روشمند مصطلح می‌شود.



«روح حساب» ناظر منازعهٔ بین «الگوری‌های» نو که از نوشتن اعداد بهره می‌برند و «چرتکه‌اندازان» سنتی با جداول شمارشی است[۳]. .

سخن کوتاه، معنی لغوی آن روش مخصوص حل دسته‌ای از مسائل است. واژهٔ «مسئله» به معنای پرسشی که به دنبال پاسخی برای آن هستیم. موارد زیر دو مثال از مسئله هستند.

- مرتب کردن افزایشی فهرستی از چند عدد
- بررسی حضور عددی در فهرست داده شده

مسئله دارای متغیرهایی است که مقدار معینی نگرفته‌اند و معروف به پارامتر هستند.

- پارامترهای مرتب‌سازی: فهرست و طول آن

[۳] برگفته از مروارید فلسفه Margarita Philosophica، اثر گرگور رایش، ۱۵۰۴ مسیحی.

- پارامترهای جستجو: فهرست و طول آن و عدد منظور

به دلیل داشتن پارامتر، هر مسئله دارای رده‌ای است که که هر یک با تخصیص مقداری به پارامتر تعیین می‌گردد. هر تخصیص خاص را یک نمونه می‌خوانیم. راه‌حل نمونه‌ای از یک مسئلهٔ پاسخ به پرسش مسئله در رابطه با آن نمونه است.

نمونه‌ای از مثال نخست

- فهرست $S = [10, 12, 1, 5, 14, 15, 7]$ و $n = 7$. پاسخ آن $[1,5,7,10, 12,14, 15]$

نمونه‌ای از مثال دوم

- فهرست $S = [10, 12, 1, 5, 14, 15, 7]$ و $n = 7$ و x=5. پاسخ آن «آری، x در S موجود است»

حال فرض کنید فهرست دارای هزاران عضو باشد. لزوما به شیوهٔ معمول نمی‌توان پاسخ را یافت. همچنین شیوه اقتضایی انسانی را نمی‌توان در رایانه اعمال کرد. جهت ایجاد برنامه رایانه‌ای که نمونه‌هایی از مسئله‌ای را حل کند، نیاز به تدوین رویهٔ عمومی و گام به گام جهت پاسخ‌دهی به هر نمونه داریم. رویه گام به گام را الگوریتم خوانیم و اظهار می‌کنیم که الگوریتم مسئله را حل می‌کند.

مثال- الگوریتم مثال دوم. با شروع از عضو اول فهرست x را با هر عضو فهرست مقایسه کن تا x یافته شود یا فهرست به اتمام رسد. اگر یافت شد پاسخ آری وگرنه پاسخ نه. الگوریتم بالا دارای معایب زیر است:

- مشکل بودن نگارش الگوریتم پیچیده مانند شیوه بالا
- همچنین سختی آگاهی خواننده از آن حتی با وجود نوشتن الگوریتم
- عدم امکان ایجاد توصیف برنامه‌ای از آن (البته با ابزارهای جدید تا حد خوبی ممکن شده است).

در نوشتهٔ حاضر از شبه‌کد و زبان پایتون برای نمایش الگوریتم استفاده می‌شود. پس تاکنون تعریفی شهودی از الگوریتم به دست داده شد.

تعریف غیرصوری الگوریتم در رایانش «دستورالعملی است که مراحل مختلف انجام کاری را به زبان دقیق و با جزئیات کافی به نحوی بیان کند که ترتیب مراحل روشن و شرط خاتمهٔ عملیات نیز آشکار باشد.»

پس، الگوریتم مجموعه‌ای متناهی از مراحل، غیرمبهم، اجراپذیر با شرط خاتمه است.

تمرین- آیا «حدس» الگوریتم است؟

زمینه‌های مطالعه الگوریتم: الگوریتم و مطالعهٔ آن به دلایل زیر انجام می‌پذیرد.

- طراحی الگوریتم: اعم از استقراء ریاضی، تقسیم و غلبه، حریصانه، برنامه‌نویسی پویا، پسگرد، انشعاب و تحدید. البته می‌توان نگاه به طراحی الگوریتم را به دو صورت دید، یا طبقه‌بندی بر اساس فنون همچون برنامه‌نویسی پویا و یا توجه به مسئله‌ای خاص مانند مرتب‌سازی و پیمایش گراف.
- اعتبارسنجی (اثبات درستی الگوریتم‌ها): الگوریتم در صورتی درست است که به ازای هر ورودیْ (هر نمونهْ) خروجی متناظر درست را پاسخ دهد.
- تحلیل الگوریتم (تحلیل مقدم، ارزیابی کارایی): الگوریتم در زمان اجرا روی پردازنده از حافظه جهت ذخیره‌سازی برنامه و داده‌ها استفاده می‌کند. تحلیل الگوریتم به فرایندی اطلاق می‌شود که میزان پیچیدگی زمان و پیچیدگی فضا اجرای الگوریتم را بررسی می‌کند.
- پیاده‌سازی
- آزمون برنامه شامل اشکال‌زدایی و پروفایلینگ برنامه (اندازه‌گیری کارایی و تحلیل موخر)

- پروفایلینگ: اجرای صحیح برنامه بر مجموعه داده ورودی و تعیین زمان و فضای لازم برای محاسبه نتیجه

الگوریتم‌ها

- بازگشتی: هرگاه الگوریتم خود را فراخواند. شامل دو نوع مستقیم و غیرمستقیم است.
- غیربازگشتی

مثال فیبوناتچی- بازگشتی مستقیم

```
Alg. Fib(n):
        if n <= 1:
                return n
        else:
                return Fib(n – 1) + Fib(n – 2)
```

مثال- ب‌م‌م دو عدد صحیح و مثبت و a>b>0

```
Alg. BMM(a, b):
        If b == 0:
                return a
        else:
                return BMM(b, a % b)
```

مثال- جستجوی کلید x در آرایه X[0,...,n-1]

```
Alg. Jostejo(X, n, i):
        if i >= n:
                return Hich
        else if X[i] == x:
                return i
        else:
                return Jostejo(X, n, i + 1)
```

در حالی که حالت غیربازگشتی جستجو به صورت زیر است

مثال- یافتن بازگشتی بیشِ آرایه X[0,...,n-1]

```
Alg. Bish(X, n, i):
        if i < n - 1:
                j = Bish( X, n, i + 1)
                if X[i] > X[j]:
                        k = i
                else:
                        k = j
        else:
                k = n
        return k
```

تمرین- الگوریتم بازگشتی یافتن بیش آرایه را تحلیل کنید.

تمرین- شبه کد غیربازگشتی یافتن کلید در آرایه و همچنین یافتن بیش آرایه را بنویسید و با نمونه‌های بازگشتی مقایسه کنید.

الگوریتم کارآ و تحلیل الگوریتم‌ها

مثال- جستجوی ترتیبی در مقابل جستجوی دودوئی

در ادامه به یکی از مسائل مشهور الگوریتمی را بررسی می‌کنیم. گاهی اوقات دنبال یافتن مقداری هستیم. به فرایند مذکور «جستجو» گویند. از روش‌های زیر جهت جستجو در فهرست استفاده می‌شود.

```
adadha = [123, 45, 32, 76, 89, 45, 33, 90, 178]

adadha.index(45)
1
```

اندیس‌های آغاز و پایان بازهٔ جستجو را می‌توان تعیین کرد.

```
adadha = [123, 45, 32, 76, 89, 45, 33, 90, 178]
```

```
adadha.index(45, 4, len(adadha))
5
```

ایراد دستور بالا این است که در صورتی که مقدار در مجموعه نباشد خطا برمی‌گرداند. همچنین، از عملگرهای in و not in نیز جهت تعیین وجود مقداری در فهرست می‌توان بهره برد.

ناگفته پیداست جستجو در در بین رکوردها امری شایع در کارهای رایانه‌ای است. به به عنوان مثال، شماره دانشجویی جستجو می‌شود تا ریزنمرات شخص موردنظر بازیابی شود. در مثال‌های قبلی استفاده از توابع پیش ساختهٔ پایتون را جهت جستجو نام برده شد. در ادامه سعی بر پیاده‌سازی تابع آنها روی فهرست‌ها می‌کنیم. الگوریتم پایین فهرستی و عددی را به نام کلید در ورودی دریافت می‌کند و سعی می‌کند کلید مدنظر را از بین اعداد فهرست جستجو کند و در صورتی که موجود باشد اندیس متناظرش را برگرداند.

```
// yaftan mqdar andis ba halqe tekrar
#include <stdio.h>

int jostejo_khatti(int* arr, int n, int klid) {

   // halqe tekrar  baraye jostejoye mqdr klid
   for (int i = 0; i < n; i++) {

      // bargasht klid dar sorat yaftan
      if (arr[i] == klid) {
         return i;
      }
   }

   // bargasht mqdari jahat elame nabod klid dar arraye
   return -1;
}
```

```
int main() {
   int arr[] = { 10, 50, 30, 70, 80, 60, 20, 90, 40 };
   int n = sizeof(arr) / sizeof(arr[0]);
   int klid = 9;

   // frakhani jostejo_khatti()
   int i = jostejo_khatti(arr, n, klid);

   // chap mqdare brgashti ba jostejo_khatti()
   if (i == -1)
      printf("mqdar klide %d mojod nist", klid);
   else
      printf("mqdar klide %d dar andise: %d", klid, i);

   return 0;
}
```

در الگوریتم جستجوی اخیر در بدترین حالت به تعداد ورودی‌ها جستجو انجام خواهد شد. اما اگر فهرست مرتب باشد و مقادیر از کوچک به بزرگ مرتب باشند می‌توان بدترین حالت را به میزان زیادی کاهش داد. یکی از الگوریتم‌های مهم جستجو که دارای بدترین زمان جستجوی کمتری باشد «الگوریتم جستجو دودوئی» است. در الگوریتم مذکور در ابتدا تعداد فهرست مشخص و سپس بر اساس آن اندیس عدد میانه را تعیین می‌کند. مقدار کلید با عدد متناظر اندیس میانه مقایسه می‌شود. اگر برابر باشند، اندیس برگردانده می‌شود و اگر کوچکتر باشد در نیمه چپ فهرست و اگر کلید بزرگتر باشد در نیمهٔ راست فهرست جستجو ادامه خواهد یافت. عمل مذکور در هر نیمه ادامه خواهد یافت. در واقع هر دفعه جستجو در نیمی از فهرست ادامه می‌یابد. می‌توان نشان داد که در بدترین حالت لگاریتمی در مبنای دو مقایسه انجام خواهد پذیرفت. به عنوان مثال اگر طول فهرست ۱۰۰۰ باشد، در جستجوی معمولی در

بدترین حالت ۱۰۰۰ مقایسه، ولی در جستجوی درختی حداکثر ۱۰ مقایسه انجام خواهد یافت که بهبودی بسیار مناسب است. در ادامه جستجوی دودوئی را معرفی می‌کنیم.

```
// yaftan mqdar andis dar arraye morattab
#include <stdio.h>
int jostejo_dodoei(int* arr, int n, int klid) {
   // halqe tekrar  baraye jostejoye mqdr klid
   int chap = 0;
   int rast = n - 1;
   int vst = -1;
    while(chap <= rast) {
            int vst = (chap + rast) / 2;
             if(klid == arr[vst]){
                return vst;
             }
             else if(klid < arr[vst]){
                rast = vst - 1;
             }
             else{
                chap =  vst + 1;
             }
   }
   // bargasht mqdari jahat elame nabod klid dar arraye
   return -1;
}
int main() {
   int arr[] = { 3, 5, 7, 9, 13, 17, 20, 30, 40, 60, 70, 80 };
   int n = sizeof(arr) / sizeof(arr[0]);
   int klid = 95;
   // frakhani jostejo_dodoei()
   int i = jostejo_dodoei(arr, n, klid);

   // chap mqdare brgashti ba jostejo_dodoei()
   if (i == -1)
```

```
    printf("mqdar klide %d mojod nist", klid);
  else
    printf("mqdar klide %d dar andise: %d", klid, i);
  return 0;
}
```

پس، در ابتدا دو اندیس چپ و راست که معادل اندیس اولین مقدار و آخرین مقادیر فهرست را تعریف می‌کنیم. تا زمانی که چپ‌ترین اندیس از اندیس راست کوچکتر یا مساوی باشد حلقه را ادامه می‌دهیم. در هر تکرارِ حلقهْ اندیس میانی برابر میانگین جز صحیح دو اندیس قبلی تعریف می‌شود. اگر کلید بزرگتر از مقدار متناظر اندیس میانی باشد، پرچم اندیس چپ برابر با مقدار بعدی اندیس میانی قرار می‌گیرد و اگر کلید کوچکتر باشد مقدار اندیس راست برابر با اندیس قبل از اندیس میانی قرار می‌گیرد. در غیر این صورت مقدار کلید و مقدار متناظر اندیس میانی برابرند، پس اندیس میانی برگشت داده می‌شود. در صورتی که حلقه به پایان برسد و اندیسی در حلقه برگردانده نشود به معنای نبود مقدار کلید در فهرست است. در نتیجه پیام نبود مقدار ایجاد می‌گردد.

در مثال قبل دیدیم که فهرست مرتب چه تاثیر بزرگی بر بهینگی الگوریتم جستجو می‌گذارد. جدول مقایسه تعداد مقایسه‌ها در دو روش جستجو ترتیبی و جستجو دودوئی را مشاهده کنید. توجه کنید به دلیل استفاده از ساختار فهرست (لیست) پایتون در عنوان اندازه فهرست نوشته شده است ولی در حالت کلی اندازه آرایه نوشته می‌شود.

اندازه فهرست	تعداد مقایسه‌های جستجو ترتیبی	تعداد مقایسه‌های جستجو دودوئی
۱۲۸	۱۲۸	۸
۱۰۲۴	۱۰۲۴	۱۱
۱۰۴۸۵۷۵	۱۰۴۸۵۷۵	۲۱
۴۴۲۹۴۹۶۷۲۹۶	۴۴۲۹۴۹۶۷۲۹۶	۳۳

بنابراین سعی می‌شود که ساختار داده‌های چند مقداری مرتب باشند تا بتوان جستجو دودوئی را روی آنها اعمال کرد. در نتیجه سعی بر مرتب‌سازی چند مقداری‌ها می‌شود. همچنین، مرتب‌سازی تاثیر فراوان در دانش و فن رایانه داشته است. در مطالب آینده چند نوع الگوریتم مرتب‌سازی معرفی می‌شوند. تفاوت آنها در میزان استفاده بهینه منابع سیستم شامل حافظه و زمان است.

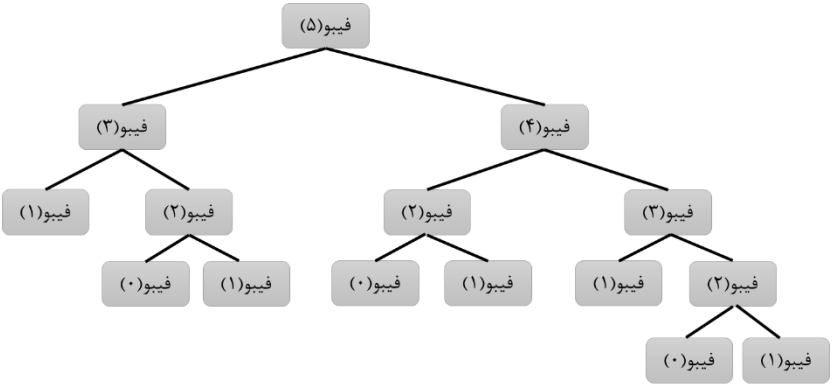
در ادامه بررسی و تحلیل الگوریتم مثال تابع فیبوناتچی را بررسی می‌کنیم. عدد فیبوناتچی را به صورت زیر و بازگشتی حساب می‌کنیم:

```
// Chap adade Fibonacci- bazgashti
#include <stdio.h>

// Tabe bazgashti
int fibo_bazgashti(int n) {
   if (n <= 1) {
      return n;
   } else {
      return fibo_bazgashti(n - 1) + fibo_bazgashti(n - 2);
   }
}

int main() {
   int n;
   printf("Adade: ");
   scanf("%d", &n);

   for( int i = 0; i <= n; i++){
     printf("Adade Fibonacci(%d) = %d\n", i, fibo_bazgashti(i));
   }
   return 0;
}
```

برنامه بالا زمان زیادی را می‌گیرد و بسیاری از مقادیر را چندین بار محاسبه کرد. به سخن دیگر، تابع بازگشتی لزوما در همه‌جا دارای بهینگی زمان اجرا نیست.

مقدار n	0	۱	۲	۳	۴	۵	۶
تعداد فراخوانی تابع	۱	۱	۳	۵	۹	۱۵	۲۵

درخت بازگشتی الگوریتم بالا در شکل زیر رسم شده است.



قضیه- تعداد موارد بازگشتی در درخت متناظر با الگوریتم بازگشتی فیبوناتچی

$$T(n) > ۲^{\frac{n}{۲}}, n \geq ۲$$

اثبات- تمرین با استقرا

می‌توان تابع فیبوناتچی را به صورت زیر نیز نوشت تا از مشکل حاصل از برنامهٔ بازگشتی جلوگیری کرد. روش زیر تکراری است.

```
// Chap adade Fibonacci- araye
#include <stdio.h>
// Tabe
int fibo2(int n) {
   if (n <= 1) {
```

```
    return n;
  }
        int fibo[n];
        fibo[0] = 0;
        fibo[1] = 1;
        for( int i = 2; i <= n; i++){
          fibo[i] = fibo[i - 1] + fibo[i-2];
        }
  return fibo[n];
}

int main() {
  int n;
  printf("Adade: ");
  scanf("%d", &n);
  for( int i = 0; i <= n; i++){
     printf("Adade Fibonacci(%d) = %d\n", i, fibo2(i));
  }
  return 0;
}
```

می‌توان از استفاده از فهرست نیز اجتناب کرد. پس برنامه فیبوناتچی را به صورت زیر بازنویسی می‌کنیم.

```
// Chap adade Fibonacci- araye
#include <stdio.h>
// Tabe
int fibo2(int n) {
  if (n <= 1) {
    return n;
  }

  int fibo_qbli1 = 0;
  int fibo_qbli2 = 1;
  int fibo;
  for( int i = 2; i <= n; i++){
```

```
    fibo = fibo_qbli1 + fibo_qbli2;
    fibo_qbli1 = fibo_qbli2;
    fibo_qbli2 = fibo;
  }
  return fibo;
}
int main() {
  int n;
        printf("Adade: ");
  scanf("%d", &n);

        for( int i = 0; i <= n; i++){
                printf("Adade Fibonacci(%d) = %d\n", i, fibo2(i));
        }
  return 0;
}
```

تمرین- زمان اجرای دو روش را بررسی کنید. فرض هر مقدار فیبوناتچی زمان $۱۰^{-۹}$ ثانیه داشته باشد.

الگوریتم بازگشتی فیبوناتچی و الگوریتم جستجو دودوئی از نوع تقسیم و غلبه هستند. الگوریتم تکراری فیبوناتچی از نوع برنامه‌ریزی پویاست. از چنین مثال‌هایی روشن می‌شود که فنون متفاوت و آشنایی با آن نقش مهم در طراحی الگوریتم پیدا می‌کند. هر یک در بعضی مسائل شیوه‌ای مناسبتر نسبت به دیگر شیوه‌ها است.

تحلیل الگوریتم-پیچیدگی زمانی

جهت تعیین میزان کارایی الگوریتم در حل مسئله، نیاز به تحلیل الگوریتم داریم. در کار قبلی به صورت غیرصوری کار تحلیل را پیش بردیم. روش صوری آن استفاده از تحلیل پیچیدگی است.

تحلیل الگوریتم نیازمند موارد زیر است:

- تعیین عملیات مورداستفاده شامل چهار عمل اصلی، دستورات مقایسه‌ای، خواندن و نوشتن، فراخوانی پردازه‌ها، دستورات انتساب
- تعیین مجموعه داده جهت بررسی همه حالات اعم از بهترین، متوسط، و بدترین زمان اجراء

در تحلیل الگوریتمْ تحلیل مقدم متفاوت از تحلیل موخر است. مورد متقدم به دنبال تعیین کران تابعی زمان اجراست و در مورد متاخر آمارهای مصرف زمان و حافظه را گزارش می‌کند. در درس جاری منظور از تحلیل مورد متقدم است. بنابراین ساده‌سازی‌هایی را مدنظر قرار می‌دهیم. ذخیره و بازیابی هر مقدار با دیگر مقادیر برابر است. زبان برنامه‌نویسی را وارد ماجرا نمی‌کنیم. افزودگی‌های مقداردهی‌های اولیه یا خواندن و نوشتن را در نظر نمی‌گیریم. در واقع هنگام بررسی کارایی الگوریتم به تعداد حقیقی چرخه‌های پردازنده توجه نمی‌کنیم. زیرا تعداد چرخه‌ها به رایانه خاص مرتبط است. علاوه بر آن به دنبال شمارش تمامی عملیات‌های انجام یافته نیز نیستیم. صرفا به دنبال آنهایی هستیم که از رایانه، زبان، و برنامه‌نویسی مستقل است. در مثال قبل دیدیم که اندازه مسئله روی پاسخ تاثیر دارد. پس روش معمول استفاده از اندازه ورودی است. به دیگر سخن، روش معیار در تحلیل الگوریتم این است که زمان اجرای الگوریتم با اندازه ورودی افزایش می‌شود و کل زمان اجرا تقریبا متناسب با تعداد اجرای عملیات پایه (مثلا مقایسه) است. در نتیجه تحلیل کارایی الگوریتم با تعیین تعداد دفعاتی که عملیات پایه‌ای بر اساس تابعی از اندازه ورودی تحلیل می‌شود. صرفا سعی بر تحلیل مسئله بر اساس اندازه ورودی می‌شود. مثلا اگر تعداد ورودی‌های

مسئله n باشد سعی بر گزارش تابعی از میزان زمان اجرا بر اساس مقدار مذکور می‌شود.

مثال-

الف	ب	ج
x = x + y	for i ∈ n: x = x + y	for i ∈ n: for j ∈ n: x = x + y

در حالت الف تعداد اجرا ۱، در حالت ب تعداد اجرا برابر n، و در حالت ج تعداد اجرا برابر با n^2 است. بنابراین مرتبه اجرای الگوریتم را برابر تعداد دفعات اجرای تک تک دستورالعمل‌ها تعریف می‌شود.

همچنین انتخاب ورودی نیز خود به نوع مسئله مرتبط است. اندازه ورودی را می‌توان به صورت‌های زیر تعریف کرد.

- تعداد اعضای فهرست مثل جستجو، مرتب‌سازی. یا تعداد سطر و ستون در ضرب ماتریسی.
- اندازهٔ ورودی با لحاظ دو مقدار حاصل شود. به عنوان مثال، برای تعیین پیچیدگی الگوریتم‌های گراف دو مقدار تعداد راس و تعداد یال بررسی می‌شود.
- اندازهٔ ورودی با تعداد علائم موردنیاز تعیین شود. به عنوان مثال، تعداد بیت‌ها در فیبوناتچی بازگشتی.

پس از تعیین اندازه ورودی، عملیات یا مجموعه عملیات‌هایی را که کار الگوریتم متناسب با تعداد اجرای آنهاست را انتخاب می‌کنیم. مجموعه اخیر را عملیات پایه می‌خوانیم و تحلیل پیچیدگی زمان الگوریتم برابر با تعداد تکرار عملیات پایه بر اساس مقدار اندازهٔ ورودی است. فرض بر پیاده‌سازی بهینهٔ عملیات پایه است. روش همیشه

معینی برای انتخاب بهترین عملیات پایه وجود ندارد. بلکه حاصل تصمیم و تجربه است و به هنر تنه می‌زند. معمولا ساختار کنترلی بررسی نمی‌شود. گاهی اوقات یک تکرار از حلقه یک عملیات به حساب می‌آید. در نگاهی دقیق‌تر می‌توان به اجرای هر دستور ماشین را به عنوان عمل پایه تصور کرد. در مرتب‌سازی می‌توان دو عملیات مقایسه و عملیات تخصیص را در نظر گرفت. چون ممکن است که میزان سربار هر یک متفاوت باشد سعی بر بررسی هر دو مورد می‌شود.

چند مثال تحلیل پیچیدگی ساده را بررسی می‌کنیم. در مثال‌های مذکور تمامی داده دیده می‌شود تا عملیات و الگوریتم مدنظر انجام پذیرد. در این هنگام به دنبال یافتن مقدار زمان اجرا هستیم.

مثال- الگوریتم جمع اعداد: n تعداد اعداد. عمل اصلی جمع. فارغ از مقدار ورودی‌ها n تکرار حلقه رخ می‌دهد. پس $Z(n) = n$

مثال- مرتب‌سازی تبادلی

```
Alg. MorattabSazi_Tabadoli(X[0,…,n-1], n):
        for i ∈ [0,…, n-1]:
          for j ∈ [i+1,…,n-1]:
                if X[i] < X[j]:
                        X[i] ↔ X[j]
```

```
// mortabsazzi taevizi
#include <stdio.h>

int morattabsazi_taevizi(int arr[], int n) {

   int mvqt;
   for (int i = 0; i < (n - 1); i++){
      for (int j = (i + 1); j < n; j++){
         if (arr[i] > arr[j]){
```

```
            mvqt = arr[i];
            arr[i] = arr[j];
            arr[j] = mvqt;
         }
      }
   }
}

int main() {
   //int arr[] = {  70, 80, 60, 20, 90, 10, 50, 30, 40 };
   // int n = sizeof(arr) / sizeof(arr[0]);

   // vorodi az karbar shemel tedad o aeza
   int n;
   printf("Tedade aeza: ");
   scanf("%d",&n);

   int arr[n];
   for(int i = 0; i < n; i++)
   {
      printf("adade %d-om : ",i);
      scanf("%d",&arr[i]);
   }
   for(int i = 0; i < n; i++){
      printf("%d ", arr[i]);
   }
   printf("\n");
   //farakhani tabe morattabsazi_taevizi
   morattabsazi_taevizi(arr, n);

   // chap mqdare morattab-shode
   for (int i = 0; i < n; i++){
      printf("%d ", arr[i]);
   }
   return 0;
```

```
}
```

مقایسه عمل اصلی الگوریتم مرتب‌سازی تبادلی است.

$$Z(n) = (n-۱) + (n-۲) + (n-۳) + \cdots + ۱ = \frac{\big(n-۱\big)n}{۲}$$

ضرب ماتریس، عمل اصلی ضرب در حلقه داخلی $Z(n) = n^{۳}$

مثال- الگوریتم یافتن بیش

```
Alg. YaftanBish(X[0,…,n-1], n):
        i = 1
        for j in [2,…, n-1]:
            if  X[i] < X[j]:
                i = j
```

تمرین- چه روشی برای یافتن بیشینه فهرست با پیچیدگی کمتر پیشنهاد می‌دهید؟

مثال‌های قبلی صرفا وابسته به تعداد ورودی هستند. اما گاهی اوقات ممکن است که پیچیدگی زمانی صرفا وابسته به اندازه ورودی مسئله نباشد. بلکه ممکن است بسته به مقادیر ورودی نیز باشد. به دیگر سخن، و بسته به احتمال در موقعیتی در ابتدا در انتها یا در وسط مشاهده داده‌ها ختم شود. در بخش قبل مثال جستجو ذکر شد. الگوریتم جستجو ترتیبی از این نوع موارد است.

الگوریتم جستجوی ترتیبی:

بهترین زمان اجرا $\mathrm{z(n)} = ۱$ است. چرا؟ بدترین زمان اجرا آن برابر $z(n) = n$ است. متوسط زمان اجرا را به صورت امید ریاضی از طریق زیر بدست می‌آوریم.

$$z(n) = \sum_{k=0}^{n-۱} \left(k \times \frac{۱}{n} \right) = \frac{۱}{n} \sum_{k=۱}^{n} k = \frac{۱}{n} \frac{\big(n-۱\big)n}{۲} = \frac{n-۱}{۲}$$

در صورتی که p برابر با احتمال یافتن مقدار کلید در آرایه باشد زمان اجرای متوسط را می‌توان دقیق‌تر بدست آورد.

$$z(n) = \sum_{k=0}^{n-1} \left(k \times \frac{p}{n}\right) + n\left(1 - p\right) = \frac{p}{n}\frac{\left(n-1\right)n}{2} + n\left(1-p\right)$$
$$= n\left(1 - \frac{p}{2}\right) - \frac{p}{2}$$

تمرین- اگر $p = \frac{1}{2}$ مقدار چقدر خواهد شد؟ تفسیر آن چیست؟

در اینجا ذکر این نکته ضروری است که میانگین به تنهایی برای تصمیم‌گیری مناسب نیست. ممکن است تمامی مقادیر حول و حوش میانگین باشند و یا دارای وردائی زیاد نسبت به میانگین باشند. در طراحی الگوریتم نیز باید به این مورد توجه کرد وگرنه ممکن است موجب خسارات فراوان شود.

مرتبه

گاه اوقات محاسبه دقیق لازم نیست و محاسبهٔ کران جهت تعیین پیچیدگی زمانی کفایت می‌کند. مثال $0.1n^2 > 100n$ از مقدار ۱۰۰۰۰ به بعد تابع درجه دو میزان زمان بیشتری را مصرف خواهد کرد. یا در جدول زیر تفاوت‌ها بین درجات مختلف محسوس و ولی با بزرگترین درجه یکسان در مقادیر بزرگتر نسبتا ناچیز است و می‌توان نادیده گرفته شوند. مثال زیر میزان رشد دو چندجمله‌ای درجه دو را نمایش می‌دهد.

n	۱۰	۲۰	۵۰	۱۰۰	۱۰۰۰
$0.1n^2$	۱۰	۴۰	۲۵۰	۱۰۰۰	۱۰۰۰۰۰

$۰.۱n^۲ + n + ۱۰۰$	۱۲۰	۱۶۰	۴۰۰	۱۲۰۰	۱۰۱۱۰۰

نمادگذاری مجانبی: قبلا ذکر شد که مرتبه اجرای الگوریتم برابر تعداد دفعات اجرای تک تک دستورالعمل‌ها (دستورالعمل‌های پایه) تعریف می‌شود. تحلیل الگوریتم معمولا با استفاده از توابع مجانبی انجام می‌پذیرد. در ادامه نمادگذاری مجانبی را بررسی می‌کنیم.

نماد اُمیکرون بزرگ O (بدترین حالت زمان اجرا):

$$f(n) = O(g(n)) \Leftrightarrow \exists c, n_0 : \forall n \geq n_0\ 0 \leq f(n) \leq cg(n)$$

مثال- $f(n) = ۲n^۳ + ۲n^۲ = O(n^۳)$؟ $c = ۴$ و $n_0 = ۲$

مثال- $f(n) = ۲n^۳ + ۲n^۲ = O(n^۴)$؟ $c = ۴$ و $n_0 = ۲$

معمولا به دنبال کوچکترین کران بالایی هستیم.

قضیه- اگر $f(n) = a_m n^m + a_{m-۱} n^{m-۱} + \cdots + a_0, a_m > 0$، آن‌گاه:

$$f(n) = O(n^m)$$

تمرین- اثبات کنید.

نتایج O

- اگر تعداد اجرا دستوری در الگوریتمی $f(n)$ باشد آن‌گاه زمان اجرای آن از مرتبهٔ $O(n^m)$ است.

- اگر برنامه دارای k بخش با مرتبه بزرگی $O(nm_1)$، $O(nm_2)$، ...، $O(nm_k)$ باشد آن‌گاه مرتبه بزرگی برنامه $O(nm)$ است به طوری که $m =$ بیش$(m_1, m_2, \dots, m_k)$
- قانون جمع (حالت دوتائی)
- قانون حاصل‌ضرب: اگر $f_1(n) = O\left(g_1(n)\right)$ و $f_2(n) =$ $O(g_2(n))$، آن‌گاه $f_1(n)f_2(n) = O(g_1(n)g_2(n))$
- $O(f(n))O(g(n)) = O(f(n)g(n))$
- $f(n) = O(f(n))$
- $O(cf(n)) = O(f(n)), c > 0$
- $O(f(n)g(n)) = f(n)O(g(n))$
- $O(f(n)) + O(g(n)) = O(f(n) + g(n))$

مثال- $f(n) = (4n^3 + 5n^2 + 7n)(8logn)$ از $O(n^3 logn)$

نماد امگا بزرگ Ω (بهترین زمان اجرا)

$$f(n) = \Omega\big(g(n)\big) \Leftrightarrow \exists c, n_0 : \forall n \geq n_0\ 0 \leq cg(n) \leq f(n)$$

مثال- $f(n) = 3n^3 + 2n^2 = \Omega(n^3)$ ؟ $c = 3$ و $n_0 = 1$

همچنین $3n^2$ و $3n$ نیز در مثال اخیر صدق می‌کنند ولی معمولا به دنبال بزرگترین کران پائین هستیم.

نماد تتا بزرگ Θ (میانگین زمان اجراء)

$$f(n) = \Theta\big(g(n)\big) \Leftrightarrow \exists c_1, c_2, n_0 : \forall n \geq n_0 \; 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

قضیه $f(n) = \Theta\big(g(n)\big)$ اگر و تنها اگر $f(n) = O\big(g(n)\big)$ و $f(n) = \Omega\big(g(n)\big)$

مثال- $f(n) = 3n^3 + 2n^2 = \theta(n^3)$؟

نماد o

$$f(n) = o\big(g(n)\big) \Leftrightarrow \exists c, n_0 : \forall n \geq n_0 \; 0 \leq f(n) < cg(n)$$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

مثال- $7n^2 + 8n + 3 = o(n^3)$

مثال- $7n^2 + 8n + 3 = O(n^2)$ است ولی $7n^2 + 8n + 3 \neq o(n^2)$

مثال- $\log(n) = o(n)$

تمرین نشان دهید که الف- $n^b = o(a^n)$ ب- به ازای b>a>0 داریم $a^n = o(b^n)$ ج- $a^n = o(n!)$.

نماد ω

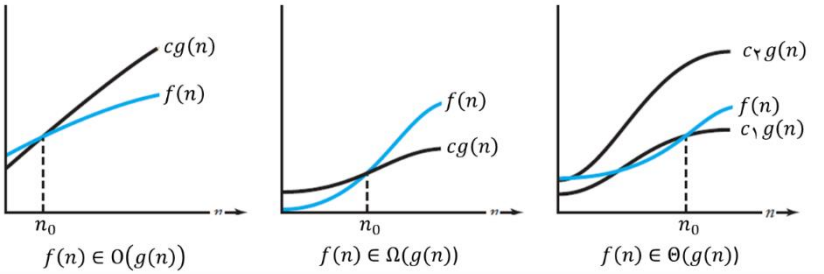
$$f(n) = \omega\big(g(n)\big) \Leftrightarrow \exists c, n_0 : \forall n \geq n_0 \; 0 \leq cg(n) < f(n)$$

قضیه $f(n) = \omega(g(n))$ اگر و فقط اگر $g(n) = o(f(n))$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

مثال- $7n^2 + 8n + 3 = \omega(n)$ است ولی $7n^2 + 8n + 3 \neq \omega(n^2)$

مثال- $n= \omega(\log n)$



مثال-

الف- $O\left(n^2\right)$

- $3n\log n + 8, 4n^2, 5n + 7, 6n^2 + 9, 2n\log n, n^2 + 2n$

ب- $\Omega\left(n^2\right)$

- $4n^2, 4n^3 + 3n^2, n^2 + 9, 6n^6 + n^4, 5n^2 + 2n, 2^n + 4n$

ج- $\Theta\left(n^2\right) = O(n^2) \cap \Omega\left(n^2\right)$

- $4n^2, 6n^2 + 9, 5n^2 + 2n$

قضیه

$$\lim_{n\to\infty}\frac{f(n)}{g(n)}=\begin{cases}c, c>0\ , f(n)=\Theta(g(n)\\ 0, f(n)=o(g(n)\\ \infty, f(n)=\omega(g(n)\end{cases}$$

مثال- تحلیل الگوریتم مرتب‌سازی حبابی

```
Alg. MorattabSazi_Hobabi(X[0,…,n-1], n):
        for i in [0,…, n-2]:
          for j in [n-1,…, i +1]:
                If X[j-1] > X[j] & i >=0:
                        (X[j − 1] ↔ X[j])
```

پیاده‌سازی

```
// mortabsazzi hobabi
#include <stdio.h>

int morattabsazi_hobabi(int arr[], int n) {

   int mvqt;
   for (int i = 0; i < (n - 1); i++){
      for (int j = 0; j < n - i - 1; j++){
         if (arr[j] > arr[j + 1]){
            mvqt = arr[j + 1];
            arr[j + 1] = arr[j];
            arr[j] = mvqt;
         }
      }
   }
}

int main() {
   //int arr[] = {  70, 80, 60, 20, 90, 10, 50, 30, 40 };
   // int n = sizeof(arr) / sizeof(arr[0]);
```

```
    // vorodi az karbar shemel tedad o aeza
    int n;
    printf("Tedade aeza: ");
    scanf("%d",&n);

    int arr[n];
    for(int i = 0; i < n; i++)
    {
       printf("adade %d-om : ",i);
       scanf("%d",&arr[i]);
    }

    for(int i = 0; i < n; i++){
       printf("%d ", arr[i]);
     }
    printf("\n");

    //farakhani tabe morattabsazi_hobabi
    morattabsazi_hobabi(arr, n);

    // chap mqdare morattab-shode
    for (int i = 0; i < n; i++){
       printf("%d ", arr[i]);
    }

    return 0;
}
```

بدترین حالت، بهترین حالت، حالت میانگین الگوریتم بالا را حساب کنید.

تحلیل الگوریتم مرتب‌سازی درجی

```
Alg. MorattabSazi_Darji(X[0,…,n-1], n):
        for j in [1,…, n-1]:
                mqdar = X[j]
                i = j - 1
```

```
while mqdar < X[i]:
        X[i +1] = X[i]
        i = i - 1
```

$X[j+1] = mqdar$

```
// mortabsazzi darji
#include <stdio.h>

int morattabsazi_darji(int arr[], int n) {
   int klid, mvqt, j;
   for (int i = 1; i < n; i++){
      klid = arr[i];
      j = i - 1;
      while (klid < arr[j] && j >= 0){
            arr[j + 1] = arr[j];
            j = j - 1;
      }
      arr[j + 1] = klid;
   }
}

int main() {
   int arr[] = {  70, 80, 60, 20, 90, 10, -50, 30, 40 };
   int n = sizeof(arr) / sizeof(arr[0]);

   //farakhani tabe morattabsazi_darji
   morattabsazi_darji(arr, n);

   // chap mqdare morattab-shode
   for (int i = 0; i < n; i++){
      printf("%d ", arr[i]);
   }
   return 0;
}
```

صرفا کافی است تعداد دفعات اجرای دستور مقایسه‌ای شمرده شود. بدترین حالت: هنگامی که حلقه داخلی همواره اجرا شود. در صورتی که آرایه نزولی مرتب شده باشد از مرتبه $O(n^۲)$ است. بهترین حالت هنگامی است که حلقه داخلی هیچ وقت اجرا نشود یا داده‌ها به صورت صعودی مرتب شده باشند و در نتیجه از $\mathrm{O(n)}$ است. حالت میانگین آن نیز از مرتبه $O(n^۲)$ است.

مرتبه‌های بزرگی معمول به صورت زیر هستند.

$$O\left(۱\right) < O(\log n) < O(n) < O(nlogn) < O\left(n^۲\right) < O\left(n^۳\right) < O(۲^n)$$

همچنین با داشتن دو متغیر n و m رابطه $۲^n \neq O(n^m)$ همیشه برقرار است. اگر کران پایین از مرتبهٔ نمایی دو باشد الگوریتم بسیار بد عمل می‌کند و همیشه دنباله حل مسئلهٔ نمایی و تبدیل آن به مسئلهٔ چندجمله‌ای هستیم.

خاصیت تابع‌های مقایسه مرتبه

- تراگذری (تعدی)

$$f(n) = \Theta\big(g(n)\big), g(n) = \Theta\big(h(n)\big) \Longrightarrow f(n) = \Theta\big(h(n)\big)$$

$$f(n) = O\big(g(n)\big), g(n) = \mathrm{O}\big(h(n)\big) \Longrightarrow f(n) = \mathrm{O}\big(h(n)\big)$$

$$f(n) = \Omega\big(g(n)\big), g(n) = \Omega\big(h(n)\big) \Longrightarrow f(n) = \Omega\big(h(n)\big)$$

$$f(n) = \mathrm{o}\big(g(n)\big), g(n) = \mathrm{o}\big(h(n)\big) \Longrightarrow f(n) = \mathrm{o}\big(h(n)\big)$$

$$f(n) = \omega\big(g(n)\big), g(n) = \omega\big(h(n)\big) \Longrightarrow f(n) = \omega\big(h(n)\big)$$

- بازتابی

$$f(n) = \Theta\big(f(n)\big)$$

$$f(n) = O\big(f(n)\big)$$

$$f(n) = \Omega\big(f(n)\big)$$

- تقارن

$$f(n) = \Theta\big(g(n)\big) \Leftrightarrow g(n) = \Theta\big(f(n)\big)$$

- تقارن ترانهاده

$$f(n) = \mathrm{O}\big(g(n)\big) \Leftrightarrow g(n) = \Omega\big(f(n)\big)$$

$$f(n) = o\big(g(n)\big) \Leftrightarrow g(n) = \omega\big(f(n)\big)$$

سه علامت امیکرون و امگا و تتا را می‌توان با کوچکتر و بزرگتر و مساوی در دو مقایسه دو عدد متناظر دانست. مقایسهٔ دو عدد حقیقی a و b از سه حالت یا a=b یا a<b یا a>b خارج نیستند که به سه شاخگی معروف است. هرچند سه شاخگی برای دو عدد ممکن است ولی برای توابع لزوما برقرار نیست، مثلا چنین رابطهٔ مجانبی بین n و $n^{1+\sin n}$ وجود ندارد.

قضیه- اگر $f(n) = o(g(n))$ آن‌گاه $f(n) = O\big(g(n)\big) - \Omega\left(g(n)\right)$

تمرین- تابع $f(n) = \begin{cases} n, \text{زوج } n \\ 1, \text{فرد } n \end{cases}$ است. نشان دهید $f(n) = O(n) - \Omega(n)$ ولی $f(n) \neq o(n)$.

روش‌های تحلیل پیچیدگی مبتنی بر روابط بازگشتی

روش حدسی

مثال فاکتوریل- بازگشتی مستقیم

```
Alg. Factorial(n):
        if n <= 1:
                return n
        else:
                return n * Factorial(n – 1)
```

$$\begin{cases} z_n = z_{n-۱} + ۱ \\ z_0 = 0 \end{cases}$$

$$z_۱ = z_{۱-۱} + ۱ = z_0 + ۱ = 0 + ۱ = ۱$$

$$z_۲ = z_{۲-۱} + ۱ = z_۱ + ۱ = ۱ + ۱ = ۲$$

$$z_۳ = z_{۳-۱} + ۱ = z_۲ + ۱ = ۲ + ۱ = ۳$$

پایه استقراء

$$z_0 = 0$$

فرض استقراء

$$z_n = n$$

گام استقراء

$$z_{n+۱} = n + ۱$$

$$z_{n+۱} = z_{(n+۱)-۱} + ۱ = z_n + ۱ = n + ۱$$

مثال-جستجو دودوئی

$$\begin{cases} z_n = z_{\frac{n}{۲}} + ۱ \\ z_۱ = ۱ \end{cases}$$

$$z_۲ = z_{۲/۲} + ۱ = z_۱ + ۱ = ۱ + ۱ = ۲$$

$$z_۴ = z_{۴/۲} + ۱ = z_۲ + ۱ = ۲ + ۱ = ۳$$

$$z_۸ = z_{۸/۲} + ۱ = z_۴ + ۱ = ۳ + ۱ = ۴$$

$$z_{۱۶} = z_{۱۶/۲} + ۱ = z_۸ + ۱ = ۴ + ۱ = ۵$$

$$z_n = \log n + ۱$$

پایه

$$z_۱ = \log ۱ + ۱ = ۱$$

فرض

$$z_n = \log n + ۱$$

حکم

$$z_{۲n} = \log ۲n + ۱$$

$$z_{۲n} = z_{۲n/۲} + ۱ = z_n + ۱ = \log n + ۱ + ۱$$
$$= \log n + \log ۲ + ۱ = \log ۲n + ۱$$

مثال

$$\begin{cases} z_n = ۲z_{\frac{n}{۲}} \\ z_۱ = ۱ \end{cases}$$

$$z_۲ = ۲z_{۲/۲} = ۲z_۱ = ۲$$

$$z_۴ = ۲z_{۴/۲} = ۲z_۲ = ۲^۲$$

$$z_۸ = ۲z_{۸/۲} = ۲z_۴ = ۲^۳$$

$$z_{۱۶} = ۲z_{۱۶/۲} = ۲z_۸ = ۲^۴$$

$$z_n = ۲^{\log n}$$

تمرین- اثبات کنید.

$$z_n = ۲^{\log n} = n^{\log ۲} \approx n^{۲.۸۱}$$

مثال-

$$\begin{cases} z_n = ۲z_{\frac{n}{۲}} + n - ۱ \\ z_۱ = 0 \end{cases}$$

$$z_۲ = ۲z_{۲/۲} + ۲ - ۱ = ۲z_۱ + ۱ = ۱$$

$$z_۴ = ۲z_{۴/۲} + ۴ - ۱ = ۲z_۲ + ۳ = ۵$$

$$z_۸ = ۲z_{۸/۲} + ۸ - ۱ = ۲z_۴ + ۷ = ۱۷$$

$$z_{۱۶} = ۲z_{۱۶/۲} + ۱۶ - ۱ = ۲z_۸ + ۱۵ = ۴۹$$

عدم امکان یافتن حدسی بابت یافتن معادله بسته

حل بازگشتی با معادله مشخصه

بازگشت خطی همگن

تعریف-

$$a_0 z_n + a_۱ z_{\mathrm{n-۱}} + \cdots + a_k z_{\mathrm{n-k}} = 0$$

مثال

$$\begin{cases} z_n - ۵z_{n-۱} + ۶z_{n-۲} = 0 \\ z_0 = 0 \\ z_۱ = ۱ \end{cases}$$

$$z_n = r^n$$

$$z_n - ۵z_{n-۱} + ۶z_{n-۲} = r^n - ۵r^{n-۱} + ۶r^{n-۲} = 0$$

$$r^n - ۵r^{n-۱} + ۶r^{n-۲} = r^{n-۲}\left(r^۲ - ۵r + ۶\right) = 0$$

$$r^۲ - ۵r + ۶ = \left(r - ۳\right)\left(r - ۲\right) = 0$$

$$z_n = ۳^n, z_n = ۲^n$$

$$z_n = c۳^n + d۲^n$$

$$z_0 = c + d = 0$$

$$z_۱ = ۳c + ۲d = ۱$$

$$c = -۱, d = ۱ \Longrightarrow z_n = ۳^n - ۲^n$$

تعریف- معادلهٔ $a_0 z_n + a_۱ z_{n-۱} + \cdots + a_k z_{n-k} = 0$ دارای معادله مشخصهٔ $r^k + \alpha_۱ r^{k-۱} + \cdots + \alpha_k r^0 = 0$ است.

قضیه- معادلهٔ $a_0 z_n + a_۱ z_{n-۱} + \cdots + a_k z_{n-k} = 0$ با معادله مشخصهٔ $\alpha_0 r^k + \alpha_۱ r^{k-۱} + \cdots + \alpha_k r^0 = 0$ دارای k ریشه متمایز باشد، پاسخ بازگشتی برابر است با

$$z_n = c_۱ r_۱^n + c_۲ r_۲^n + \cdots + c_k r_k^n$$

مثال

$$\begin{cases} z_n - ۳z_{n-۱} - ۴z_{n-۲} = 0 \\ z_0 = 0 \\ z_۱ = ۱ \end{cases}$$

$$z_n = r^n$$

$$\left(r^{۲} - ۳r - ۴\right) = 0$$

$$r^{۲} - ۳r - ۴ = \left(r - ۴\right)\left(r + ۱\right) = 0$$

$$z_n = ۴^n, z_n = (-۱)^n$$

$$z_n = c۴^n + d(-۱)^n$$

$$z_0 = c + d = 0$$

$$z_۱ = ۴c - d = ۱$$

$$c = \frac{۱}{۵}, d = -\frac{۱}{۵} \Longrightarrow z_n = ۲^n$$

مثال فیبوناتچی- بازگشتی مستقیم

```
Alg. Fib(n):
        if n <= 1:
                return n
        else:
                return Fib(n – 1) + Fib(n – 2)
```

حل- فرض می‌کنیم a_n تعداد فراخوانی‌های ***Fib(n)*** باشد. آن‌گاه

$$\begin{cases} a_n = a_{n-۱} + a_{n-۲} + ۱ \\ a_0 = ۱, a_۱ = ۱ \end{cases}$$

حال $\mathrm{b_n} = a_n + ۱$ و افزودن یک به طرفین معادلات داریم:

$$\begin{cases} \mathrm{b_n} = b_{n-۱} + b_{n-۲} \\ b_0 = ۲, b_۱ = ۲ \end{cases}$$

رابطه را به دو صورت می‌توان حل کرد.

الف- از معادله مشخصه استفاده می‌کنیم.

$$r^۲ - r - ۱ = 0 \Longrightarrow r_۱, r_۲ = \frac{۱ \pm \sqrt{۵}}{۲}$$

پس

$$b_n = c_۱ \left(\frac{۱ + \sqrt{۵}}{۲} \right)^n + c_۲ \left(\frac{۱ - \sqrt{۵}}{۲} \right)^n$$

با اعمال شرایط مرزی داریم:

$$b_0 = ۲ \Longrightarrow ۲ = c_۱ + c_۲$$

$$b_۱ = ۲ \Longrightarrow ۲ = c_۱ \left(\frac{۱ + \sqrt{۵}}{۲} \right) + c_۲ \left(\frac{۱ - \sqrt{۵}}{۲} \right)$$

در نتیجه

$$c_۲\left(\frac{۱-\sqrt{۵}}{۲}\right) \text{ و } c_۱ = -\left(\frac{۱-\sqrt{۵}}{۲}\right)$$

و

$$b_n = \frac{۲}{\sqrt{۵}}\left(\frac{۱+\sqrt{۵}}{۲}\right)^{n+۱} - \frac{۲}{\sqrt{۵}}\left(\frac{۱-\sqrt{۵}}{۲}\right)^{n+۱}$$

و

$$a_n = \frac{۲}{\sqrt{۵}}\left(\frac{۱+\sqrt{۵}}{۲}\right)^{n+۱} - \frac{۲}{\sqrt{۵}}\left(\frac{۱-\sqrt{۵}}{۲}\right)^{n+۱} - ۱$$

در نهایت

$$a_n = ۲F_{n+۱} - ۱$$

F_n عدد n-ام فیبوناتچی است.

ب- با استفاده از سری مولد

$$\begin{cases} \mathrm{a_n} = a_{n-۱} + a_{n-۲} + ۱ \\ a_0 = ۱, a_۱ = ۱ \end{cases}$$

سری مولد را به صورت زیر تعریف می‌کنیم:

$$A(x) = \sum_{n=0}^{\infty} a_n x^n$$

$$A(x) = a_0 + a_1 x + \sum_{n=2}^{\infty} a_n x^n$$

$$A(x) = 1 + x + \sum_{n=2}^{\infty} \left(a_{n-1} + a_{n-2} + 1\right) x^n$$

$$A(x) = \frac{1 - x + x^2}{(1 - x - x^2)(1 - x)}$$

قضیه- اگر معادله مشخصه بازگشتی خطی همگن دارای ریشه‌ای مضاعف باشد، آن گاه

$$z_n = r^n, z_n = nr^n, z_n = n^2 r^n, \dots, , z_n = n^{m-1} r^n$$

مثال- حل بازگشتی

$$z_n - 7z_{n-1} + 15z_{n-2} - 9z_{n-3} = 0$$

$$z_0 = 0, z_1 = 1, z_2 = 2$$

۱- یافتن معادله مشخصه

$$r^3 - 7r^2 + 15r - 9 = 0$$

۲- حل معادله مشخصه

$$r^{۳} - ۷r^{۲} + ۱۵r - ۹ = \left(r - ۱\right)\left(r - ۳\right)^{۱} = 0$$

۳- اعمال قضیه

$$z(n) = a۱^{n} + b۳^{n} + cn۳^{n}$$

۴- استفاده از مقدار اولیه جهت تعیین ضرائب

$$z_0 = 0 = a۱^{0} + b۳^{0} + c(0)۳^{0}$$

$$z_{۱} = ۱ = a۱^{۱} + b۳^{۱} + c\left(۱\right)۳^{۱}$$

$$, z_{۲} = ۲ = a۱^{۲} + b۳^{۲} + c\left(۲\right)۳^{۲}$$

$$a + b = 0$$

$$a + ۳b + ۳c = ۱$$

$$a + ۹b + ۱۸c = ۲$$

۵- معادله نهایی

$$z(n) = (-1)(1^n) + (1)(3^n)\left(-\frac{1}{3}\right)(n3^n)$$
$$= -1 + 3^n - n3^{n-1}$$

تمرین- قضیه را اثبات کنید.

تمرین-

$$z(n) = 5z(n-1) - 7z(n-2) + 3z(n-3), n > 2$$

$$z(0) = 1, z(1) = 2, z(2) = 3$$

بازگشتی خطی ناهمگن

تعریف-

$$a_0 z_n + a_1 z_{n-1} + \cdots + a_k z_{n-k} = f(n)$$

تاکنون الگوریتم حل معادلهٔ ناهمگن ابداع نشده است. پس صرفا توابعی خاصی را بررسی می‌کنیم. مثال زیر از موارد خاص است.

$$a_0 z_n + a_1 z_{n-1} + \cdots + a_k z_{n-k} = b^n p(n)$$

حالت خاص اخیر را می‌توان با تبدیل آن به بازگشتی خطی همگن حل کرد. مثال زیر را ببینید.

مثال- $z_n - ۳z_{n-۱} = ۴^n, z_0 = 0, z_۱ = ۴$

۱- جانشینی n با $n - ۱$

$$z_{n-۱} - ۳z_{n-۲} = ۴^{n-۱}$$

۲- تقسیم دو طرف معادله اصلی بر ۴

$$\frac{z_n}{۴} - \frac{۳z_{n-۱}}{۴} = ۴^{n-۱}$$

۳- دو معادله بخش ۱ و ۲ دارای سمت راست یکسان هستند. پس سمت چپ معادله‌ها هم با هم برابر است و در نتیجه داریم،

$$\frac{z_n}{۴} - \frac{۷z_{n-۱}}{۴} + ۳z_{n-۲} = 0$$

یا

$$z_n - ۷z_{n-۱} + ۱۲z_{n-۲} = 0$$

امکان حل با معادله بازگشتی خطی همگن

$$r^۲ - ۷r + ۱۲ = \left(r - ۳\right)\left(r - ۴۴\right) = 0$$

$$z(n) = a۳^n + b۴^n$$

با اعمال شرایط اولیه داریم

$$z(n) = +۴^{n+۱} - ۴\left(۳^n\right)$$

پس دارای دو جواب 3^n و 4^n است. پاسخ نخست از جواب همگن حاصل شده است. پاسخ دوم از بخش ناهمگن معادله بازگشتی حاصل می‌شود.

قضیه معادله بازگشتی خطی ناهمگن به صورت

$$a_0 z_n + a_1 z_{n-1} + \cdots + a_k z_{n-k} = b^n p(n)$$

را می‌توان به معادله خطی باگشتی همگن با معادله مشخصی زیر

$$\left(a_0 r^k + a_1 r^{k-1} + \cdots + a_k r^0\right)(r-b)^{d+1} = 0$$

تبدیل کرد که d درجه $p(n)$ است. پس معادله مشخصه دارای دو بخش معادله مشخصه متناظر بازگشت همگن و ضریب حاصل از بخش ناهمگن معادله بازگشتی است.

مثال -

$$z(n) - 3z\left(n-1\right) = 4^n\left(2n+1\right), n > 1$$

$$z(0) = 0, z\left(1\right) = 12$$

۱- یافتن معادله مشخصه از بخش همگن معادله

$$z(n) - 3z\left(n-1\right) = 0 \Longrightarrow r - 3 = 0 \Longrightarrow r = 3$$

۲- یافتن بخش متناظر ناهمگن معادله

$$(r-b)^{d+1} = \left(r-۴\right)^{۱+۱} = \left(r-۴\right)^{۲}$$

۳- اعمال قضیهٔ اخیر و حل معادله

$$\left(r-۳\right)\left(r-۴\right)^{۲} = 0$$

۴- اعمال قضیه قبلی

$$z(n) = a۳^{n} + b۴^{n} + cn۴^{n}$$

۵- با بررسی شرایط اولیه

$$z(n) = ۲۰\left(۳^{n}\right) - ۲۰\left(۴^{n}\right) + ۸\left(n۴^{n}\right)$$

تمرین- $\mathrm{z(n)} - \mathrm{z}\left(\mathrm{n}-۱\right) = \mathrm{n}-۱, z(0) = 0$

تمرین- قضیه را اثبات کنید.

تغییر متغیر (تبدیل دامنه)

مثال-

$$Z(n) = Z\left(\frac{n}{۲}\right) + ۱, Z\left(۱\right) = ۱$$

$$n = ۲^{k}, k = \log n$$

$$Z\left(۲^{k}\right) = Z\left(\frac{۲^{k}}{۲}\right) + ۱ = Z\left(۲^{k-۱}\right) + ۱$$

$$z_k = Z\left(۲^{k}\right)$$

$$z_k = z_{k-۱} + ۱$$

$$z_k = k$$

$$Z\left(۲^{k}\right) = z_k = k \Longrightarrow Z(n) = \log n$$

$$Z(n) = ۱ + logn$$

مثال-

$$Z(n) = ۲Z\left(\frac{n}{۲}\right) + n - ۱, Z\left(۱\right) = 0$$

$$n = ۲^{k}, k = \log n$$

$$Z\left(۲^{k}\right)=۲Z\left(\frac{۲^{k}}{۲}\right)+۲^{k}-۱=۲Z\left(۲^{k-۱}\right)+۲^{k}-۱$$

$$z_k=Z\left(۲^{k}\right)$$

$$z_k=۲z_{k-۱}+۲^{k}-۱$$

$$z_k=۲^{k}+k۲^{k}$$

$$Z\left(۲^{k}\right)=۲^{k}+k۲^{k}=k\Longrightarrow Z(n)=n+n\log n$$

$$Z(n)=nlogn-(n-۱)$$

تمرین-

$$Z(n)=۷Z\left(\frac{n}{۲}\right)+۱۸\left(\frac{n}{۲}\right)^{۲},Z\left(۱\right)=0$$

حل معادلات بازگشتی با جانشینی

در صورتی که نتوان با روش‌های قبل به پاسخ رسید می‌توان از صنعت جانشینی بهره برد.

مثال-

$$z_n=z_{n-۱}+\frac{۲}{n},z_۱=0$$

$$z_n=z_{n-۱}+\frac{۲}{n}$$

$$z_{n-۱} = z_{n-۲} + \frac{۲}{n-۱}$$

$$z_{n-۲} = z_{n-۳} + \frac{۲}{n-۲}$$

$$\vdots$$

$$z_{۲} = z_{۱} + \frac{۲}{۲}$$

$$z_{۱} = 0$$

$$z_n = z_{n-۱} + \frac{۲}{n}$$

$$= z_{n-۲} + \frac{۲}{n-۱} + \frac{۲}{n}$$

$$= z_{n-۳} + \frac{۲}{n-۲} + \frac{۲}{n-۱} + \frac{۲}{n}$$

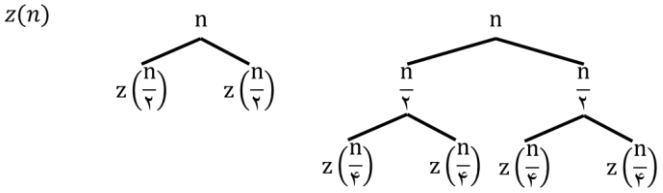
$$\vdots$$

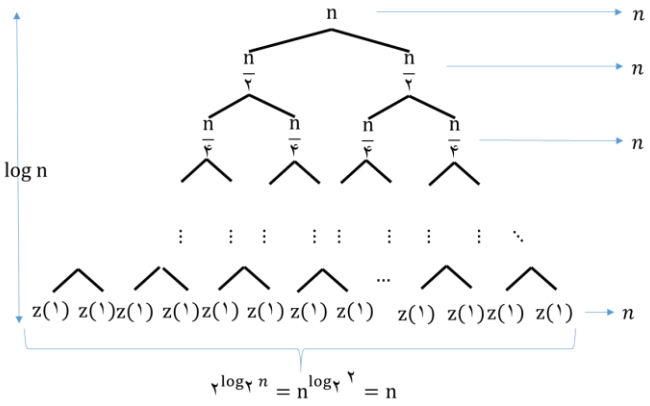
$$= 0 + \frac{۲}{۲} + \frac{۲}{n-۲} + \frac{۲}{n-۱} + \frac{۲}{n}$$

$$= ۲\sum_{i=۲}^{n}\frac{۱}{i} = ۲\ln n$$

روش درختی

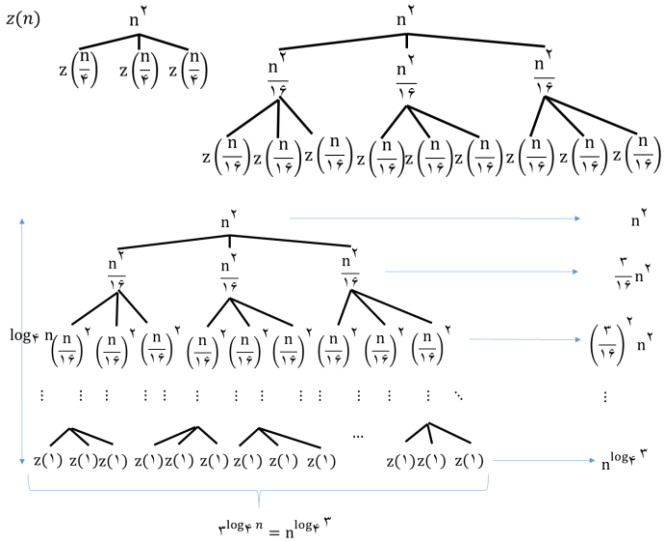
$$z(n) = ۲z\left(\frac{n}{۲}\right) + \theta(n)$$





$$۲^{\log_۲ n} = n^{\log_۲ ۲} = n$$

$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$



$$z(n) = n^{۲} + \frac{۳}{۱۶}n^{۲} + \left(\frac{۳}{۱۶}\right)^{۲} n^{۲} + \cdots + \left(\frac{۳}{۱۶}\right)^{\log_{۴} n} n^{۲} + \theta\left(n^{\log_{۴} ۳}\right)$$

$$= \sum_{i=0}^{\log_{۴} n} \left(\frac{۳}{۱۶}\right)^{i} n^{۲} + \theta\left(n^{\log_{۴} ۳}\right)$$

$$< \sum_{i=0}^{\infty} \left(\frac{۳}{۱۶}\right)^{i} n^{۲} + \theta\left(n^{\log_{۴} ۳}\right)$$

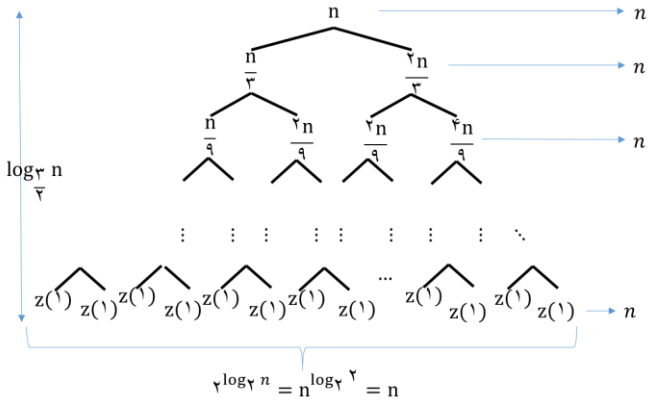
$$= \frac{۱}{۱ - \left(\frac{۳}{۱۶}\right)} n^{۲} + \theta\left(n^{\log_{۴} ۳}\right)$$

$$= \frac{۱۶}{۱۳} n^{۲} + \theta\left(n^{\log_{۴} ۳}\right)$$

$$= O\left(n^{۲}\right)$$

مثال نامنظم

$$T(n) = T\left(\frac{n}{۳}\right) + T\left(\frac{۲n}{۳}\right) + \theta(n)$$



$$nlogn$$

قضیه اصلی حل روابط تقسیم و غلبه. فرض کنیم که

$$\begin{cases} z(n) = az\left(\frac{n}{b}\right) + cn^k \\ t\left(۱\right) = \theta\left(۱\right) \end{cases}$$

که a و b و c اعداد مثبت و $a \geq ۱$ و $b > ۱$، آن‌گاه

الف- اگر $b^k < a$ آن‌گاه $z(n) = \theta\left(n^{\log_b a}\right)$

ب- اگر $b^k = a$ آن‌گاه $z(n) = \theta\left(n^k \log n\right)$

چ- اگر $b^k > a$ آن‌گاه $z(n) = \theta\left(n^k\right)$

اثبات-

نکته: درصورت جانشینی $z(n) = az\left(\frac{n}{b}\right) + cn^k$ با $z(n) \leq at\left(\frac{n}{b}\right) + cn^k$ ($z(n) \geq at\left(\frac{n}{b}\right) + cn^k$) نتایج با O یا Ω جانشین خواهد شد.

مثال- $z(n) = ۸z\left(\frac{n}{۴}\right) + ۵n^۲$

$$۸ < ۴^۲ \Longrightarrow z(n) \in \Theta\left(n^۲\right)$$

مثال- $z(n) = ۹z\left(\frac{n}{۳}\right) + ۵n^۱$

$$۹ > ۳^۱ \Longrightarrow z(n) \in \Theta\left(n^{\log_۳ ۹}\right) = \Theta\left(n^۲\right)$$

مثال- $z(n) = ۲z\left(\frac{n}{۲}\right) + ۱$

مثال- $z(n) = ۲z\left(\frac{n}{۲}\right) + n$

مثال- $z(n) = z\left(\frac{۲n}{۳}\right) + ۱$

مثال- $z(n) = ۳z\left(\frac{n}{۳}\right) + n^{۲}$

قضیه عمومی حل روابط تقسیم و غلبه

فرض کنیم که معادله بازگشتی زمان اجرای الگوریتمی عبارت است از

$$\begin{cases} z(n) = az\left(\frac{n}{b}\right) + f(n), n = b^m \\ z\left(۱\right) = \theta\left(۱\right) \end{cases}$$

که a و b و c اعداد مثبت و $a \geq ۱$ و $b > ۱$، C>0 و f(n)>0 آن‌گاه

الف- اگر $f(n) = o\left(n^{\log_b a}\right)$ آن‌گاه $z(n) = \theta\left(n^{\log_b a}\right)$

ب- اگر $f(n) = \theta\left(n^{\log_b a}\right)$ آن‌گاه $z(n) = \theta\left(n^{\log_b a} \log n\right)$

چ- اگر $f(n) = \omega\left(n^{\log_b a}\right)$ آن‌گاه $z(n) = \theta\left(f(n)\right)$

مثال- $z(n) = ۱۰۰z\left(\frac{n}{۹۹}\right) + \log n!$

مثال- $z(n) = ۸z\left(\frac{n}{۹}\right) + n \log n$

مثال- $z(n) = ۴z\left(\frac{n}{۲}\right) + ۵n^{۲} + nlog\ n + ۶n$

لم- اگر در رابطه $z(n) = az\left(\frac{n}{b}\right) + f(n)$ داشته باشیم $f(n) = \theta\left(n^{\log_b a}\log^{j} n\right)$ آن‌گاه $z(n) = \theta\left(n^{\log_b a}\log^{j+۱} n\right)$

تمرین- اثبات کنید.

بازگشت اکرا-باتزی

$$z(n) = f(n) + \sum_{i=۱}^{k} a_i z(\frac{n}{b_i})$$

$$z(n) = \theta(n^p) + \theta\left(n^p \int_{n'}^{n} \frac{f(x)}{x^{n+۱}} dx\right)$$

پیوست مقدمات ریاضی

خاصیت یکنوائی:

- تابع $f(n)$ افزایشی یکنوا است اگر $m \leq n$ آن‌گاه $f(m) \leq f(n)$.
- تابع $f(n)$ کاهشی یکنوا است اگر $m \leq n$ آن‌گاه $f(m) \geq f(n)$.
- تابع $f(n)$ اکیدا افزایشی یکنوا است اگر $m < n$ آن‌گاه $f(m) < f(n)$.
- تابع $f(n)$ اکیدا کاهشی یکنوا است اگر $m < n$ آن‌گاه $f(m) > f(n)$.

سقف و کف

- $\lfloor n \rfloor = n = \lceil n \rceil$
- $x - 1 < \lfloor x \rfloor \leq \mathrm{x} \leq \lceil x \rceil < x + 1$
- $-\lceil x \rceil = \lfloor -x \rfloor$ یا $-\lfloor x \rfloor = \lceil -x \rceil$
- $\left\lceil \frac{n}{۲} \right\rceil + \left\lfloor \frac{n}{۲} \right\rfloor = n$
- $\left\lceil \frac{\lceil \frac{n}{a} \rceil}{\mathrm{b}} \right\rceil = \left\lceil \frac{\mathrm{n}}{\mathrm{ab}} \right\rceil$
- $\left\lfloor \frac{\lfloor \frac{n}{a} \rfloor}{\mathrm{b}} \right\rfloor = \left\lfloor \frac{n}{ab} \right\rfloor$
- $\left\lceil \frac{a}{\mathrm{b}} \right\rceil \leq \frac{\left(a + \left(b - ۱\right)\right)}{b}$
- $\left\lfloor \frac{a}{b} \right\rfloor \geq \frac{\left(a + \left(b - ۱\right)\right)}{b}$
- $\lfloor n + x \rfloor = n + \lfloor x \rfloor$
- $\lceil n + x \rceil = n + \lceil x \rceil$

ریاضی پیمانه

- $a\%n = a - n \left\lfloor \frac{a}{n} \right\rfloor$
- $0 \leq a\%n < n$

نمایی‌ها

- $\lim_{\mathrm{n} \to \infty} \frac{n^b}{a^n} = 0$
- $n^b = o(a^n)$
- $e^x = ۱ + x + \frac{x^۲}{۲} + \cdots + \frac{x^n}{n!} + \cdots$
- $۱ + x \leq e^x$
- اگر $|\mathrm{x}| \leq ۱$، آن‌گاه $۱ + x \leq e^x \leq ۱ + x + x^۲$

- $e^x = ۱ + x + \theta\left(x^۲\right)$
- $\lim_{n\to\infty}\left(۱ + \frac{x}{n}\right)^n = e^x$

لگاریتم

- $a = b^{\log_b a}$
- $\log_b a = \frac{\log_c a}{\log_c b}$
- $\log_b\left(\frac{۱}{a}\right) = -\log_b a$
- $\log_b a = \frac{۱}{\log_a b}$
- $a^{\log_b n} = n^{\log_b a}$
- اگر $x < y$ آن‌گاه $\log x < \log y$

دوجمله‌ای‌ها

- $(x+y)^n = \sum_{r=0}^{n}\binom{n}{r}x^{n-r}y^r$
- $\binom{n}{r} = \binom{n}{n-r}$
- $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$
- $\sum_{r=0}^{k}\binom{m}{r}\binom{n}{k-r} = \binom{m+n}{k}$

فاکتوریل‌ها

- $n! = o(n^n)$
- $n! = \omega\left(۲^n\right)$
- $\log(n!) = \Theta(n\log n)$

- $n! = \sqrt{۲\pi n}\left(\frac{n}{e}\right)^n \left(۱ + \theta\left(\frac{۱}{n}\right)\right)$
- $n! = \sqrt{۲\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n}$ به طوری که $\frac{۱}{۱۲n+۱} < \alpha_n < \frac{۱}{۱۲n}$
- معادلهٔ استرلینگ $n! \cong \sqrt{۲\pi n}\left(\frac{n}{e}\right)^n$

سری‌ها

$$\sum_{r=0}^{n} r = \frac{n\left(n+۱\right)}{۲}$$

$$\sum_{r=0}^{n} a^r = \frac{a^{n+۱} - ۱}{a - ۱}$$

$$\sum_{r=0}^{\infty} a^r = \frac{۱}{۱ - a}, 0 < a < ۱$$