«بسمه تعالی»

ساختمان داده ها و الگوریتم ها

الگوریتمهای مرتب سازی آرایه ها

Array Sorting Algorithms

تهیه کننده :
مسلم رحیمی


کجا بذارمش؟!

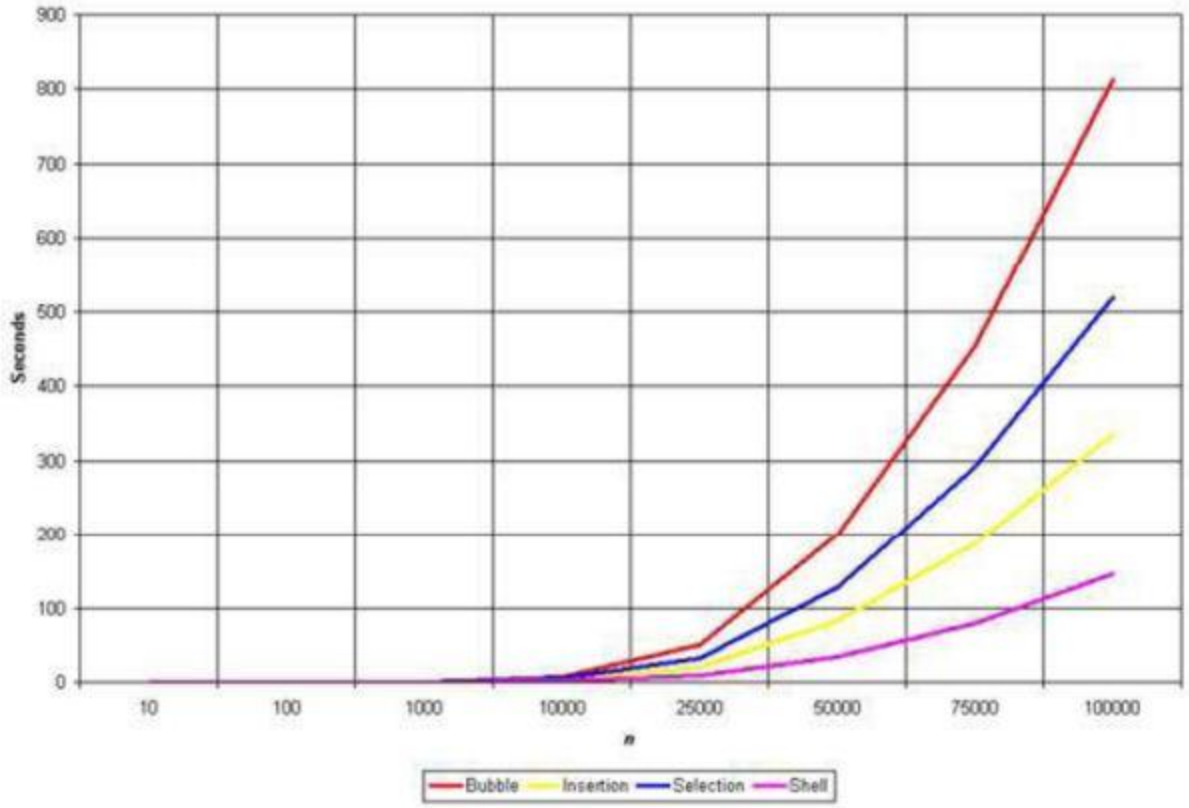
فهرست

الگوریتم	بهترین B(n)	میانگین A(n)	بدترین W(n)	حافظه*	پایدار	روش عمومی
حبابی Bubble	n	n^2	n^2	1	بله	جابجایی
درجی Insertion	n	n^2	n^2	1	بله	درجی
انتخابی Selection	n^2	n^2	n^2	1	خیر	انتخابی
شل Shell	n	$nlog^2n$	$nlog^2n$	1	خیر	درجی
ادغامی Merge	nlogn	nlogn	nlogn	بستگی* دارد	بله	ادغامی
هرمی Heap	nlogn	nlogn	nlogn	1	خیر	انتخابی
سریع Quick	nlogn	nlogn	n^2	Log n	بستگی دارد	جزءبندی

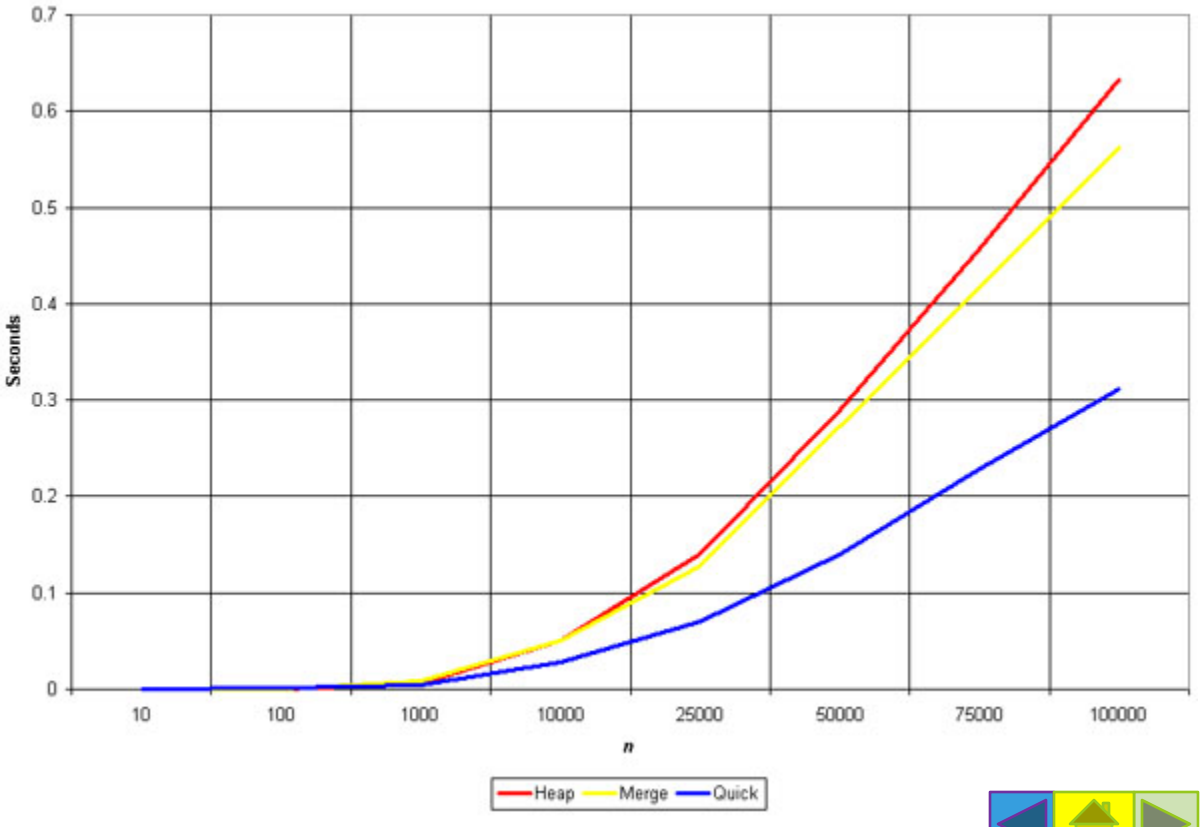
***حافظه : بیانگر مقدار حافظه کمکی که علاوه بر حافظه اشغال شده توسط خود داده ها،مورد استفاده الگوریتم قرار میگیرد.**

*** پایدار بودن آن به پیاده سازی بستگی دارد.**

مقایسه سرعت الگوریتم های: حبابی ، انتخابی ، درجی ، شل



مقایسه سرعت الگوریتم های: هرمی ، ادغامی ، سریع



1- مرتب سازی حبابی(Bubble)

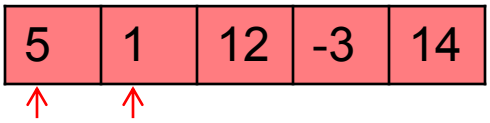
ساده ترین روش مرتب سازی الگوریتم ها به شمار می آید.در این روش هر عنصر با عنصر بعدی اش مقایسه میشود. در صورتی که عنصر دومی کوچکتر از عنصر اولی باشد، جای دو عنصر با هم عوض میشود. برنامه به کارش ادامه میدهد و عناصر دوم و سوم را با هم مقایسه میکند و این کار را تا اخر آرایه ادامه میدهد. دوباره الگوریتم ، پویش را از اول آرایه شروع میکند و مراحل قبل را تکرار میکند و این مراحل آنقدر تکرار میشوند تا آرایه کاملا مرتب شده باشد، در نتیجه بازده پایین می آید.

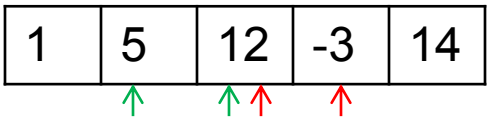
مرتبه ی پیچیدگی این الگوریتم $O(n^2)$است.

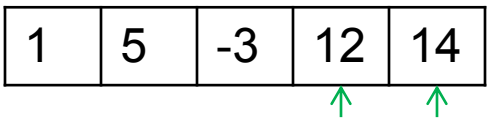
*یکی از معایب این الگوریتم ، تعداد مقایسه های زیاد آن میباشد و اگر آرایه، عناصر زیادی داشته باشد،وقت زیادی را برای مرتب کردن عناصر از برنامه و CPU خواهد گرفت.

تحلیل الگوریتم :

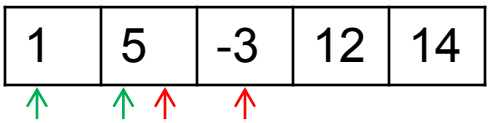
مرحله اول

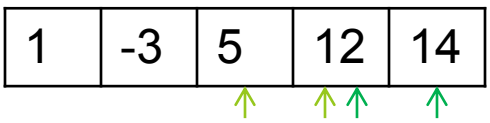




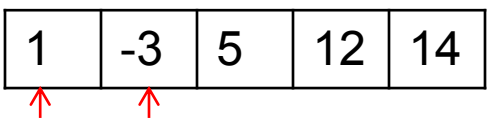


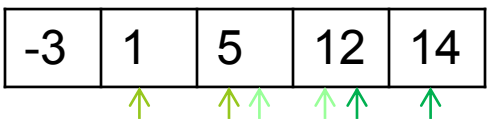
مرحله دوم

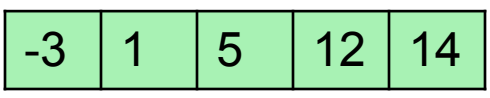




مرحله سوم







کد الگوریتم Bubble Sort به زبان C++

```
void bubble (int x[],int y)
{
int temp;
for (int i=y-1;i>0;i--)
   for (int j=0;j<i;j++)
      if (x[j] > x[j+1])
      {
        temp = x[j];
        x[j] = x[j+1];
        x[j+1] = temp;
      }
}
```

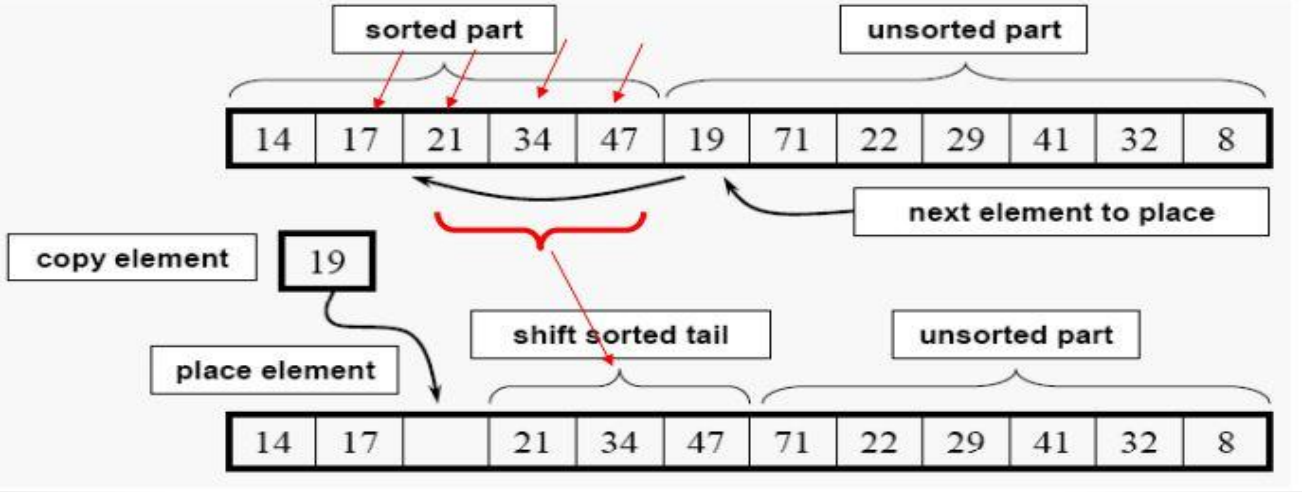
2- مرتب سازی درجی (Insertion)

این الگوریتم نیز تقریبا مانند الگوریتم حبابی عمل میکند با این تفاوت که در این روش مقایسه از عنصر دوم شروع میشود،این الگوریتم عنصر اول و دوم را با هم مقایسه میکند و در صورت نیاز مرتب میشوند و سپس سومین عنصر با عناصر اول و دوم مقایسه میشود. در صورتی که عنصر سوم از اولی کوچکتر باشد به جای اولین عنصر می نشیند و عناصر قبلی به سمت راست هل داده میشوند. اگر عنصر سوم از اولی بزرگتر و از دومی کوچکتر باشد، بین آنها درج میشود و عنصر دوم به بعد یکی به سمت راست هل داده میشود.

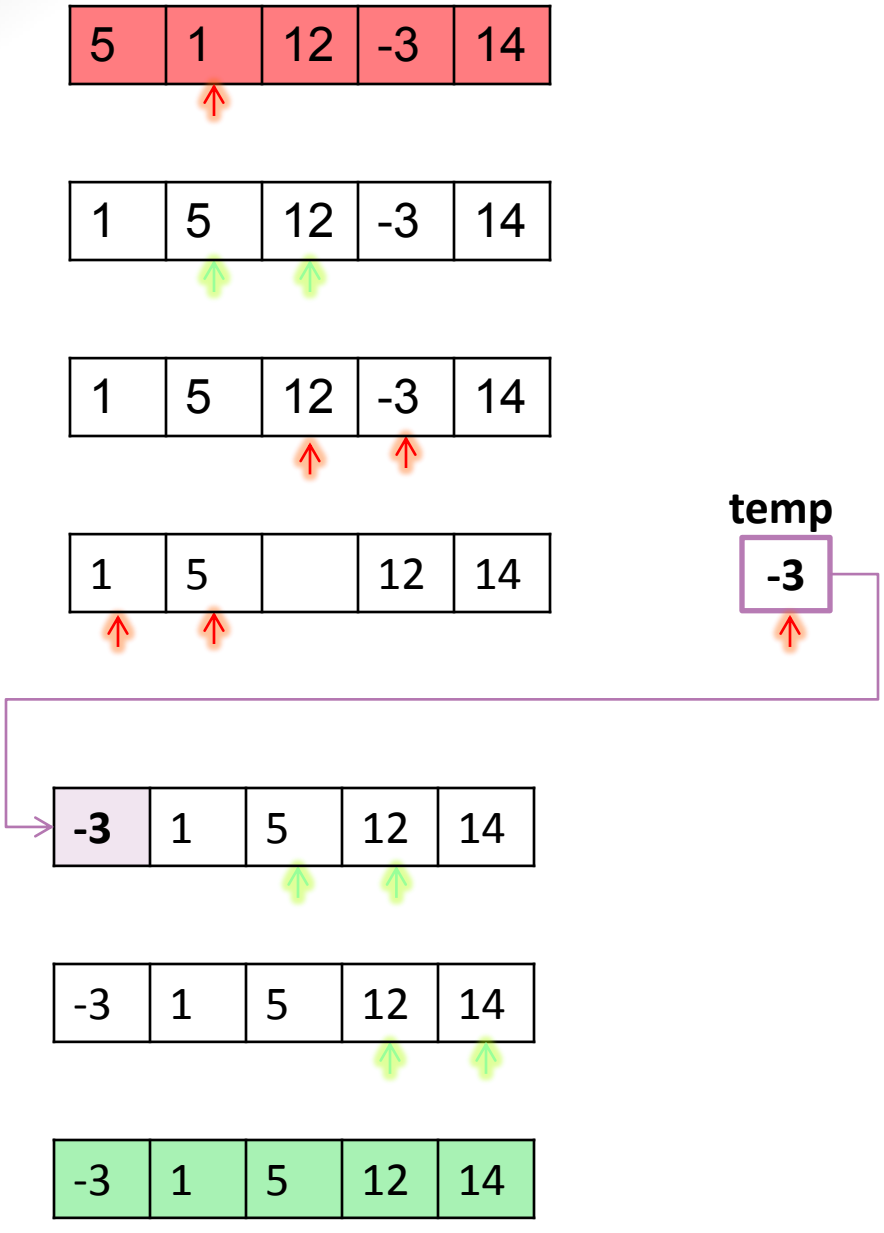
(پس در این روش همیشه عناصر ، قبل از عنصری که میخواهیم مرتبش کنیم، مرتب هستند.) این روال برای بقیه عناصر نیز اجرا میشود و هر عنصر در جای خودش قرار می گیرد تا تمام عناصر مرتب شوند.

در تکرار iام این الگوریتم ، i عنصر اول در آرایه اصلی مرتب خواهند شد.

مرتبه ی پیچیدگی این الگوریتم $O(n^2)$است.



تحليل الگوريتم :



کد الگوریتم Insertion Sort به زبان C++

```
void insertion (int x[],int n)
{
  int temp;
  for(int i=1;i<n;i++)
  {
     j=i;
     while(int j>0 && x[j-1]>x[j])
      {
            temp = x[j];
            x[j] = x[j-1];
            x[j-1] = temp;
            j--;
      }
   }
}
```

تفاوت الگوریتم درجی با حبابی :

در الگوریتم درجی حتما عناصر قبل از آن مرتب هستند اما در الگوریتم حبابی اینطور نیست در نتیجه بازده زمانی الگوریتم درجی از حبابی بیشتر است.

مهمترین حسن الگوریتم درجی ، برنامه نویسی آسان آن است اما به کندی اجرا میشود و این موضوع در هنگام مرتب کردن آرایه های بزرگ آشکار میشود.

این الگوریتم برای بازه های مرتب کارآمد است.

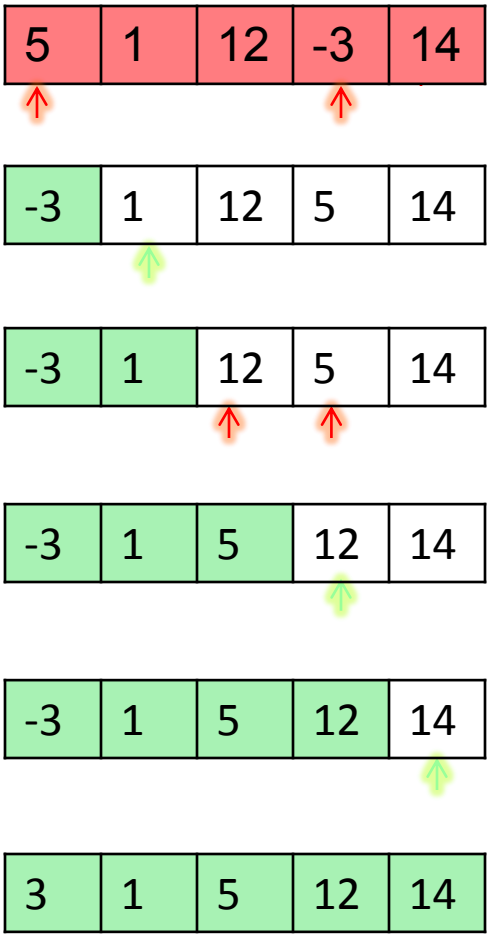
3- مرتب سازی انتخابی (Selection)

در این روش، برنامه کوچکترین مقدار را یافته و آنرا در اولین خانه ی آرایه قرار می دهد. حال که کوچکترین عضو یافت شده است، برنامه به سراغ یافتن دومین عنصر کوچک در میان اعداد باقی مانده که از 2 تا n هستند می رود و دومین عدد کوچک را در خانه دوم قرار میدهد. حال به سراغ سومین عدد کوچک می رود و این رویه را تا یافتن آخرین عدد و قرار دادن آن در جای خودش تکرار میکند. با توجه به اینکه برنامه باید nعدد را nبار با هم مقایسه کند مرتبه ی پیچیدگی این الگوریتم $O(n^2)$است.

*الگوریتم انتخابی 60% سریعتر از الگوریتم حبابی است،اما الگوریتم درجی بهتر از انتخابی و حبابی اجرا می شود.

*در الگوریتم درجی برای پیدا کردن عنصر x هر تعداد از اعداد را که نیاز است بررسی می کند ولی الگوریتم انتخابی همه عناصر باقیمانده را بررسی میکند.

تحليل الگوريتم :



کد الگوریتم Selection Sort به زبان C++

```
Void selection sort (int x [], const int n)
{
    int i , j , item ,min;
    for (i=0 ; i<n-1 ; i++)
    {
        min = I;
        for ( j=i+1 ; j<n ; j++ )
            if (x[j] < x[min])
                min = j ;
            item = x[i] ;
            x [i] = x[min] ;
            x[min] = item ;
    }
}
```

4- مرتب سازی شل (Shell)

این روش یکی از روش های مرتب سازی درجی است که بهبود یافته روش درج ساده عمل می باشد و اولین بار توسط Shell طراحی گردید.
در این روش که در چند گذر انجام می شود، درهر گذر لیست موجود، به مجموعه های مجزایی از عناصر افراز می شود و هر کدام از این قسمت ها معمولاً به روش درج ساده مرتب می شوند.
در هر گذر برای تقسیم بندی لیست به مجموعه هایی مجزا از پارامتر مشخصی مانند k استفاده می شود که پارامتر «افزایش» نامیده می شود.
به عنوان مثال اگر k=3 انتخاب شود، لیست مورد نظر به 3 مجموعه مجزا به صورت زیر تقسیم می شود:

S1= { a[0], a[3], a[6], ...}
S2= { a[1], a[4], a[7], ...}
S3= { a[2], a[5], a[8], ...}

برای اینکه عنصر i ام از s (j) یک آرایه را بدست آوریم از فرمول زیر استفاده می کنیم:

$$a\,[(i-1).k+j-1]$$

به عنوان مثال عنصر دوم از S3 برابر است با

a [(2-1)*3+3-1]= a [5]

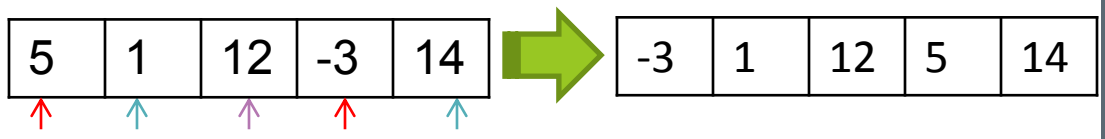
*روش درج ساده روشی است که در آن هرچه لیست اولیه مرتب تر باشد تعداد عملیات لازم جهت مرتب سازی کمتر است.
*از طرف دیگر هرچه تعداد عناصر لیست بیشتر باشد تعداد عملیات بیشتری نیاز خواهد بود.
*روش مرتب سازی شل با در نظر گرفتن این نکات سعی دارد تا اولاً روش درج ساده را با قسمت بندی عناصر به مجموعه هایی مجزا بر روی مجموعه هایی با طول کوچکتر انجام دهد.ثانیاً در هر گذر،لیست اولیه را برای مراحل بعدی مرتب تر نماید.

تحليل الگوريتم :

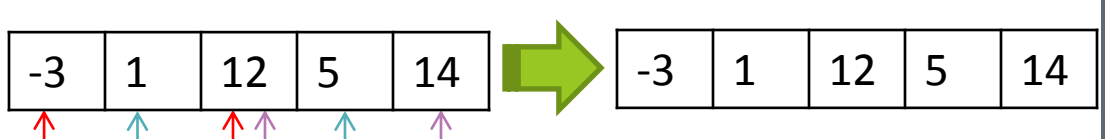
5	1	12	-3	14

فاصله مرحله

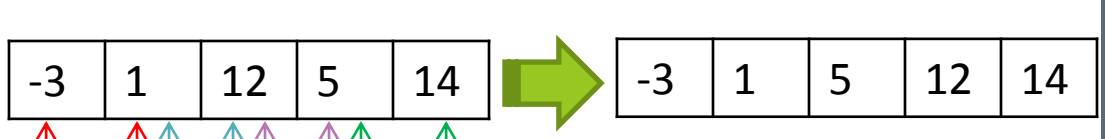
K=3 — 1



K=2 — 3



K=1 — 3



-3	1	5	12	14

کد الگوریتم Shell Sort به زبان C++

```
void shellsort(numbers[size])
{
int i,j,increment,temp;
increment=3;
while(increment>0)
{
for(i=0;i<size;i++)
{
j=i;
temp=numbers[i];
while((j>=increment) && (numbers[j-increment]>temp))
{
numbers[j]=numbers[j-increment];
j=j-increment;
}
numbers[j]=temp;
}
if(increment/2 !=0)
increment=increment/2;
else if(increment==1)
increment=0;
else
increment=1;
}
}
```

5- مرتب سازی ادغامی (Merge Sort)

این الگوریتم به روش بازگشتی(Recursive)عمل میکند و آرایه را به چند آرایه ی دو عنصری تقسیم میکند و آنها را مرتب میکند. سپس آرایه های کوچک را دوبه‌دو با هم ادغام میکند تا آرایه های مرتب 4 عنصری ایجاد شوند و بعد آرایه های 8 عنصری و به همین ترتیب پیش می رود تا آرایه اصلی بصورت مرتب شده ظاهر شود.

این روش از مراحل بازگشتی زیر تشکیل یافته است:

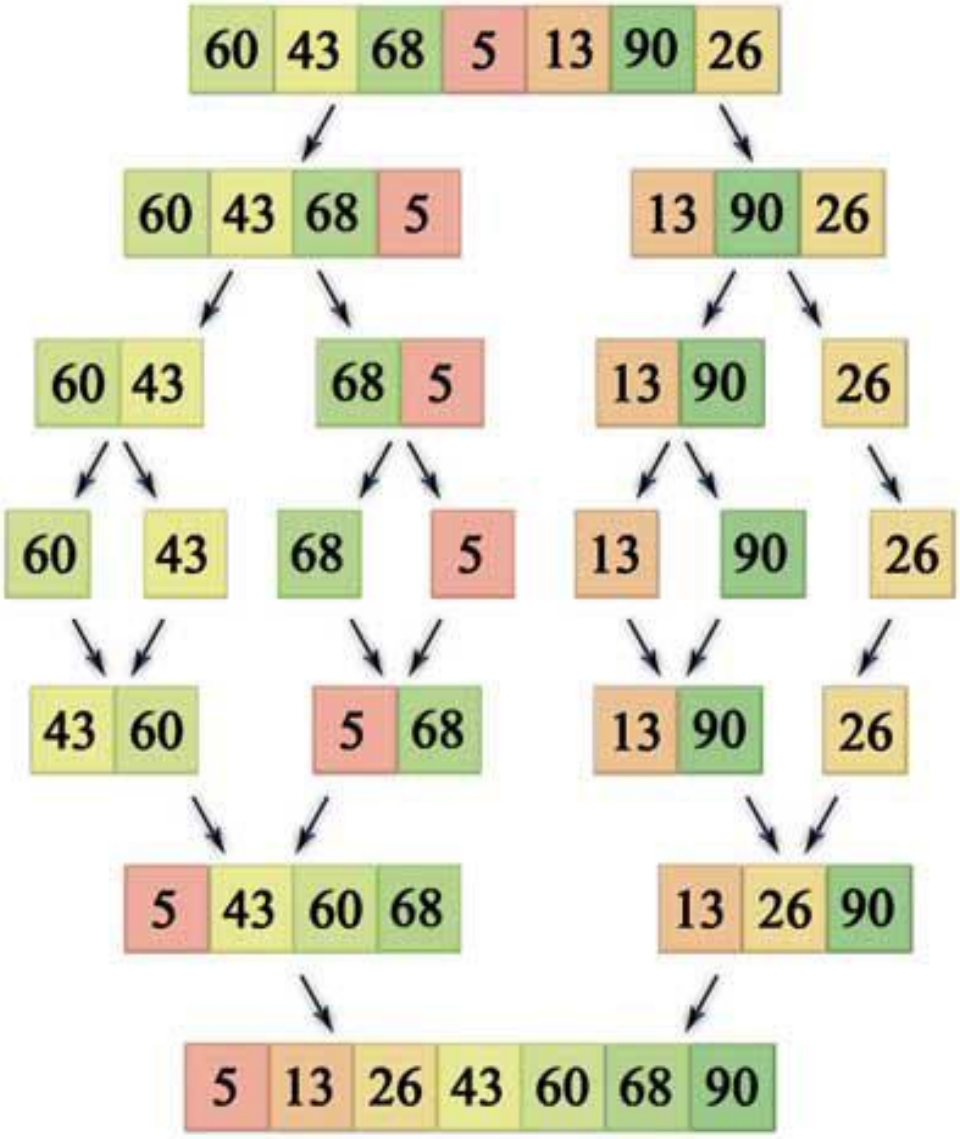
1- آرایه را به دو زیرآرایه با اندازه تقریبا یکسان تقسیم کن.

2- دو زیرآرایه را به روش مرتب‌سازی ادغامی مرتب کن.

3- دو زیرآرایه مرتب‌شده را ادغام کن.

مرتبه پیچیدگی این الگوریتم $O(n \log n)$است.

تحليل الگوريتم :



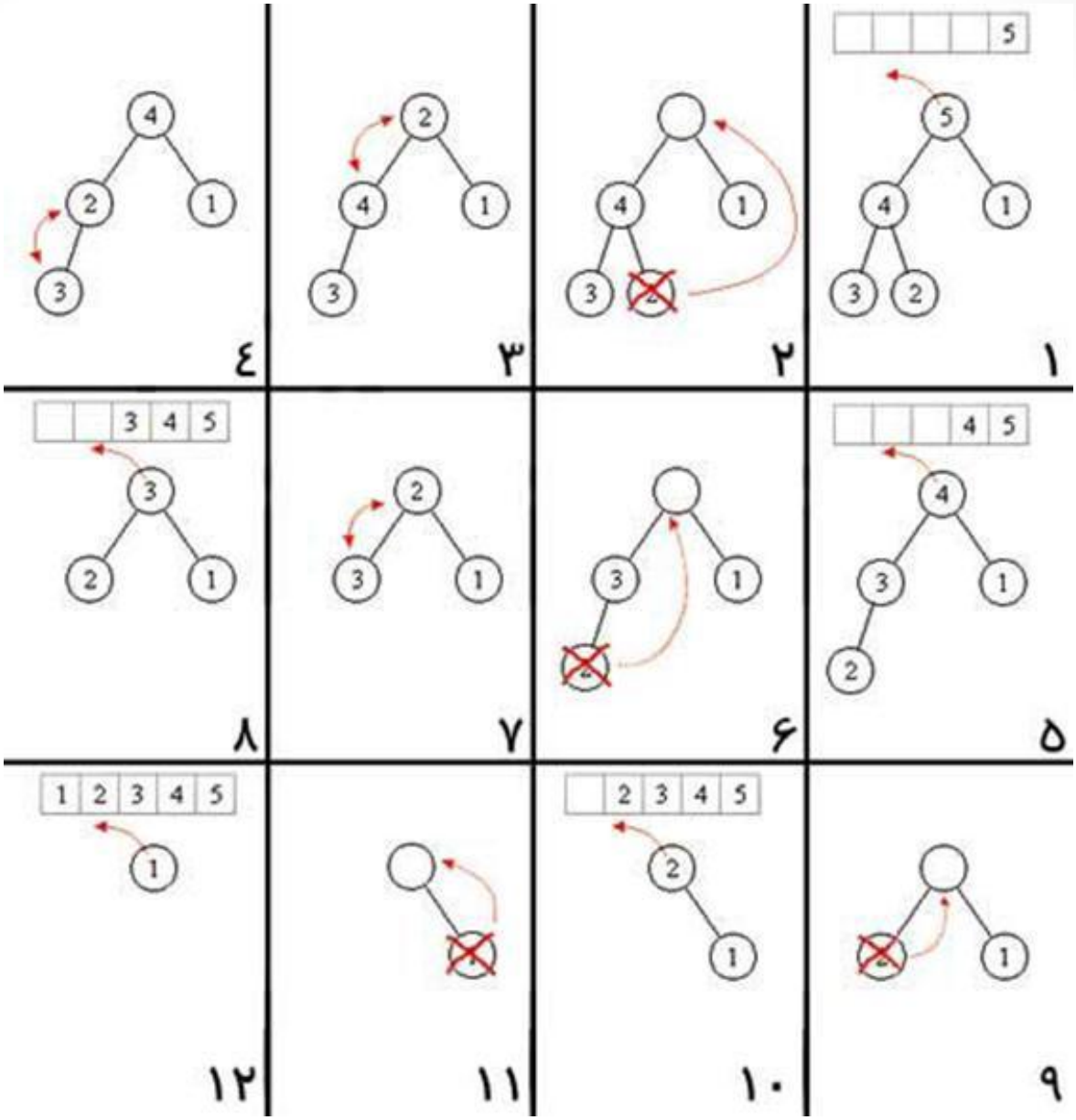
کد الگوریتم Merg Sort

```
void merge(int a[],int b[],int c[],int n,int m)
A[n];B[m];
i=1;j=1;k=0;
while(i<=n||j<=m)
do k=k+1;
if(j>m||((i<=n)&&(A[i]<=B[j])))
c[k]=a[i];
i=i+1;
else
c[k]=b[j];
j=j+1;
c[]=n+m;
return c;
```

6- مرتب سازی هرمی(Heap Sort) :

یک الگوریتم کارای مرتب سازی درجا در حالت کلی است که در بدتری حالت پیچیدگی زمانی بهینه دارد،هرچند که در عمل ، مرتب سازی سریع از آن سریعتر عمل میکند.این الگوریتم بر اساس داده ساختارهای هرم است.در این روش، برنامه از کل آرایه ی داده شده یک درخت MaxHeapمی سازد. (درخت مکس هیپ درختی دودویی و کامل است که مقدار ذخیره شده در هر گره ، بزرگتر و یا مساوی مقدار ذخیره شده در گره فرزندانش است) سپس مقدار ماکزیمم را از درخت حذف میکند و آنرا در انتهای آرایه میگذارد و دوباره از بقیه اعداد یک درخت maxHeapمیسازد و باز روش مذکور را روی آن نیز اعمال میکند تا دومین عدد بزرگ یافت شود.این کار تا زمانی که هیچ مقداری در Heap نماند و آرایه مرتب شده کامل شود،تکرار خواهد شد. در این روش آرایه از آخر به اول مرتب میشود. مرتبه پیچیدگی این الگوریتم $O(n \log n)$است.

تحليل الگوريتم :



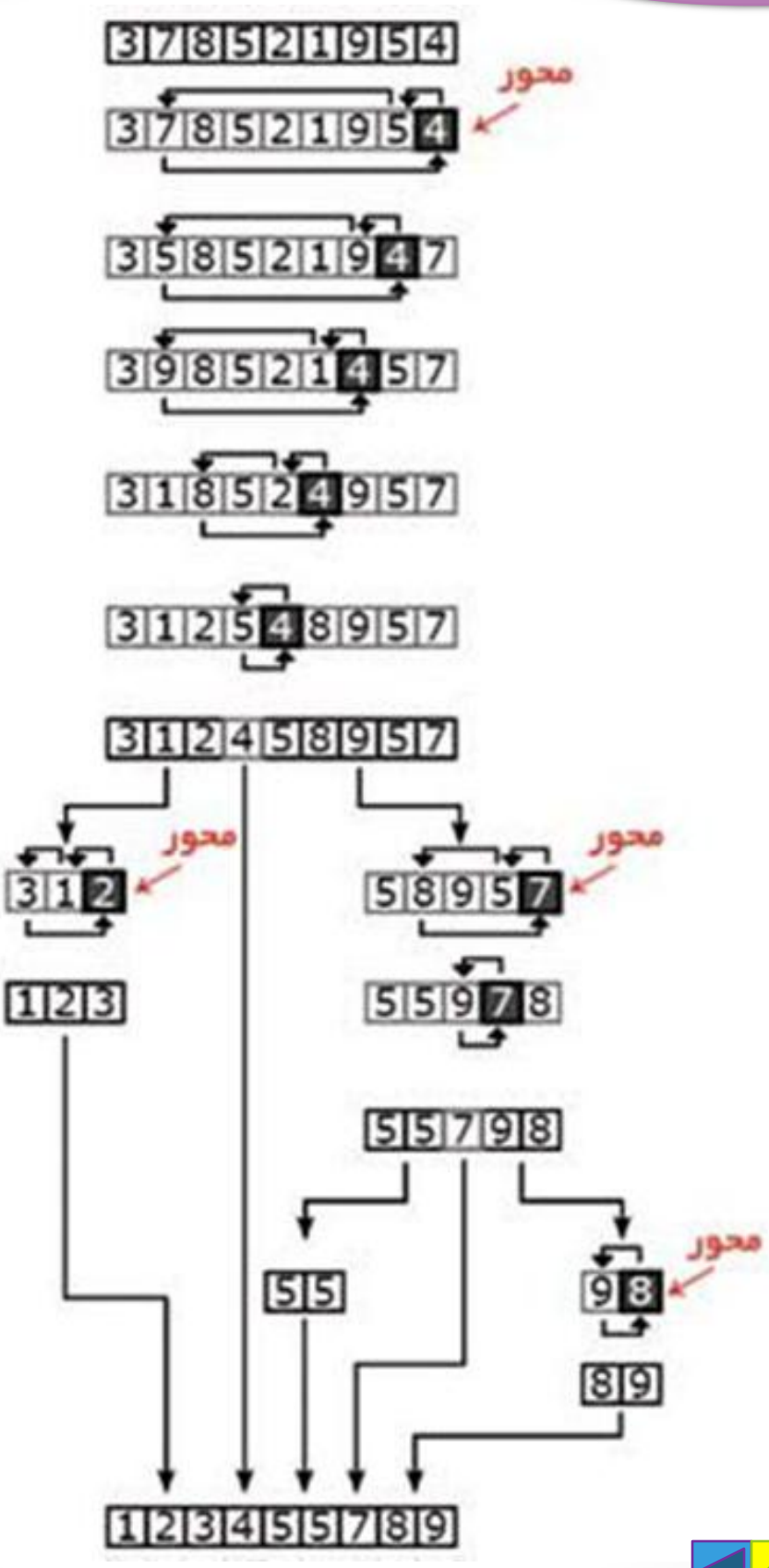
7- مرتب سازی سریع (Quick):

در این الگوریتم یک عنصر را بعنوان محور(pivot)مرتب سازی انتخاب میکنیم. و تمام عناصر کوچکتر از آن را به سمت چپ آن برده و عناصر بزرگتر را به سمت راست اش می‌بریم. حالا بخش چپ خودش یک بخش جدید است که با الگوریتمی که گفتیم آنرا مرتب میکنیم و سمت راست را نیز همینطور. یعنی در سمت چپی ها دوباره یک عنصر را بعنوان pivot در نظر میگیریم و عناصر کوچکتر از pivot را به سمت چپ آن و عناصر بزرگتر از pivotاین قسمت را ، به سمت راست pivotمی بریم. دوباره الگوریتم را روی یک چهارم های به وجود آمده اجرا میکنیم و اینکار را آنقدر ادامه میدهیم تا کل آرایه مرتب شود.

مرتبه پیچیدگی این الگوریتم در بدترین حالت $O(n^2)$است. اما در حالت میانگین $O(n \log n)$است که کمترین مرتبه پیچیدگی برای مرتب سازی اعداد به حساب می آید.

* در عمل سریعتر از الگوریتم های بهینه دیگر مرتب میکند.
* گونه ی حافظه خارجی این الگوریتم کاراست.
* پیاده سازی آن به صورت موازی هم امکان پذیر است.

تحليل الگوريتم :



کد الگوریتم Quick Sort به زبان C++

```
void quicksort(int x[],int left,int right)
{
     int i=left,j=right;
     int temp;
     int pivot=x[(left+right)/2];
     while (i<=j)
     {
          while(x[i]<pivot)
               i++;
          while(x[j]>pivot)
               j--;
          if(i<=j)
           {
              temp=x[i];
              x[i]=x[j];
              x[j]=temp;
              i++;
              j--;
           }
     }
if (left<j)
     quicksort(x,left,j);
if (i<right)
     quicksort(x,i,right);
}
```