Binary Search: جستجوی دودویی

algoritm :

تکنیکی است برای یافتن یک مقدار عددی از میان مجموعه‌ای از اعداد مرتب. این متد محدودهٔ جستجو را در هر مرحله به نصف کاهش می‌دهد، بنابراین هدف مورد نظر یا به زودی پیدا می‌شود و یا مشخص می‌شود که مقدار مورد جستجو در فهرست وجود ندارد.

جستجوی دودویی فقط در آرایه های مرتب استفاده می شود.در این روش عنصر مورد نظر با خانه وسط آرایه مقایسه می شود اگر با این خانه برابر بود جستجو تمام می شود اگر عنصر مورد جستجو از خانه وسط بزرگتر بود جستجو در بخش بالایی آرایه و در غیر این صورت جستجو در بخش پایینی آرایه انجام می شود(فرض کرده ایم آرایه به صورت صعودی مرتب شده است) این رویه تا یافتن عنصر مورد نظر یا بررسی کل خانه های آرایه ادامه می یابد..

1. get the middle element;

2. if the middle element equals to the searched value, the algorithm stops;

3. otherwise, two cases are possible:

 \*searched value is less, than the middle element. In this case, go to the step 1 for the part of the array, before middle element.

 \*searched value is greater, than the middle element. In this case, go to the step 1 for the part of the array, after middle element.

Now we should define, when iterations should stop. First case is when searched element is found. Second one is when subarray has no elements. In this case, we can conclude, that searched value doesn't present in the array.

Java :

int binarySearch(int[] array, int min, int max, int s) {

 int mid = (max – mid ) /2;

 if(s == aray[mid] )

 return mid;

 else if ( s > array[mid] )

 return binarySearch(mid, max, s);

else

 return binarySearch(min, mid, s);

}

\*--------------------------------------------------------------\*

Bubble Sort : مرتب سازی حبابی

Java :

  **public static void**bubble\_srt( **int**a[], **int**n ){
    **int**i, j,t=0;
    **for**(i = 0; i < n; i++){
      **for**(j = 1; j < (n-i); j++){
        **if**(a[j-1] > a[j]){
          t = a[j-1];
          a[j-1]=a[j];
          a[j]=t;
        }
      }
    }
  }
}

\*--------------------------------------------------------------\*

Selection Sort: مرتب سازی انتخابی

algoritm :

این الگوریتم اینگونه عمل می‌کند: ابتدا کوچکترین عنصر مجموعه اعداد را یافته با اولین عدد جابجا می کنیم.. سپس دومین عنصر کوچکتر را یافته با دومین عدد جابجا می کنیم و این روند را برای

.n-1 عدد اول تکرار می کنیم

در حقیقت در هر مرحله ما لیست خود را به دو بخش تقسیم می کنیم.زیرلیست اول که قبلاً مرتب کرده‌ایم و سایر اعضای لیست که هنوز مرتب نشده‌است. در جدول زیر مثالی از پیاده سازی این روال بر روی 6 عدد آمده‌است.

ورودی: 15 4 34 12 55 24

1) 4 15 34 12 55 24

2) 4 12 34 15 55 24

3) 4 12 15 34 55 24

4) 4 12 15 24 55 34

5) 4 12 15 24 34 55

Java:

void selectionSort(int[] a) {

 for (int i = 0; i <a.length - 1; i++) {

 int min = i;

 for (int j = i + 1; j <a.length; j++) {

 if (a[j] <a[min]) {

 min = j;

 }

 }

 int swap = a[i];

 a[i] = a[min];

 a[min] = swap;

 }

}

\*--------------------------------------------------------------\*

Insertion Sort : جست و جوی درجی

algoritm :

در مرتب سازی درجی، ابتدا عنصر دوم با عنصر اول لیست مقایسه می‌شود و در صورت لزوم با عنصر اول جابجا می‌شود به طوری که عناصر اول و دوم تشکیل یک لیست مرتب دوتایی را بدهند. سپس عنصر سوم به ترتیب با دو عنصر قبلی خود یعنی عناصر دوم و اول مقایسه و درجای مناسبی قرار می‌گیرد به طوری که عناصر اول و دوم و سوم تشکیل یک لیست مرتب سه تایی را بدهند.سپس عنصر چهارم به ترتیب با سه عنصر قبلی خود یعنی عنصرسوم و دوم و اول مقایسه و درجای مناسب قرار می‌گیرد به طوری که عناصر اول و دوم و سوم و چهارم تشکیل یک لسیت مرتب چهارتایی را بدهند و در حالت کلی عنصر امم با ام منهای یک عنصر قبلی خود مقایسه می‌گردد تا در مکان مناسب قرار گیرد به طوری که ام عنصر تشکیل یک لیست مرتب ام تایی را بدهند و این روند تا مرتب شدن کامل لیست ادامه می‌یابد.

Java :

public static void insertionSort(int[] a) {

 for (int i = 1; i < a.length; i++) {

 int num = a[i];

 boolean b = true;

 for (int j = 0; b && j < i; j++) {

 if (num <= a[j]) {

 int t = a[j];

 b = false;

 a[j] = num;

 for (int k = i; k > j; k--)

 a[k] = a[k - 1];

 a[j + 1] = t;

 }

 }

 }

 }

\*--------------------------------------------------------------\*

**تمام الگوریتم ها :**

**الگوریتم مرتب سازی انتخابی** (Selection Sort):

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 آخر عدد و قرار دادن آن در جای خودش تکرار میکند. با توجه به اینکه برنامه باید n عدد را n بار با هم مقایسه کند مرتبه ی پیچیدگی این الگوریتم O(n^2) است.

**مرتب سازی حبابی** (Bubble Sort):

 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. مرتبه ی پیچیدگی این الگوریتم

 است .O(n^2)

**مرتب سازی درجی** (Insertion Sort):

در این روش عنصر اول و دوم با هم مقایسه شده 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 هشتند.) این روال برای بقیه عناصر نیز اجرا میشود و هر عنصر در جای خودش قرار می گیرد تا تمام عناصر مرتب شوند. مرتبه ی پیچیدگی این الگوریتم O(n^2) است.

\*--------------------------------------------------------------\*