مرتب سازی

جستجو

- زمانی که یک لیست از عناصر را جستجو می کنیم ، هدف پیدا نمودن عناصری است که با مقدار مورد نظر برابر باشند.

- **روش های جستجو:**

  - جستجوی ترتیبی
  - جستجوی دودویی

جستجوی ترتیبی

❖ فرض کنید که یک آرایه و یک مقدار جستجو به نام Searchnum داشته باشیم. هدف بازیابی عنصری است که مقدار آن منطبق بر searchnum باشد. اگر این آرایه دارای n عنصر باشد، آرایه را با جستجوی مقادیر عناصر list[n-1].key، ...، list[0].key مورد بررسی قرار می دهیم تا به عنصر مورد نظر برسیم یا تمام آرایه را جستجو کنیم.

26, 5, 77, 1, 61, 11, 59, 15

جستجوی ترتیبی

```
int seqsearch( int list[ ], int searchnum, int n )
{  int i,c=0;
   for (i=0; i<n; i++)
        if(list[i]==searchnum)
            {
              c=1;
              break;
            }
   if(c==1)
         return i;
   else
         return -1;
}
```

تحلیل تابع جستجوی ترتیبی

- در بدترین حالت باید تمام عناصر آرایه با عنصر مورد جستجو مقایسه شود

$O(n)$

- در بهترین حالت اولین عنصر آرایه می تواند همان عنصر مورد جستجو باشد

$\Omega(1)$

جستجوی دودویی

❖ در جستجوی دودویی بایستی آرایه به صورت مرتب شده باشد.

2 8 10 13 17 19 20 25 30

جستجوی دودویی

```
int binsearch(int list[ ], int searchnum, int n)
{
/* search list [0], ..., list[n-1]*/
   int left = 0, right = n-1, middle;
   while (left <= right) {
       middle = (left+ right)/2;
   switch (COMPARE(list[middle],searchnum)) {
       case -1: left = middle +1;
                  break;
       case  0: return middle;
       case  1:right = middle - 1;
       }
     }
     return -1;
}
```

$O(\log_2 n)$

مرتب سازی

- دو نوع از کاربردهای مهم مرتب سازی عبارتند از :
  - کمک در امر جستجو
  - تطابق عناصر و وارده ها در یک لیست

مرتب سازی حبابی

26, 5, 77, 1, 61, 11, 59, 15

مرتب سازی حبابی

```
void bubble_sort(int list[], int n)
{  int i,j,temp;
   for (int i=1; i<n; i++)
         for (int j=0; j<n-i; j++)
            if (a[j] > a[j+1])
               {temp=list[j];
                 list[j]=list[j+1];
                 list[j+1]=temp;}
}
```

مرتب سازی انتخابی

26, 5, 77, 1, 61, 11, 59, 15

مرتب سازی انتخابی

```
void selection_sort(int list[], int n)
{  int i,j,min,indexmin,temp;
   for (i=0; i<n; i++)
         {min=list[i];
          indexmin=i;
          for(j=i+1;j<n;j++)
                if(list[j]<min)
                     {min=list[j];
                       indexmin=j;}
          temp=list[i];
          list[i]=list[indexmin];
          list[indexmin]=temp;
         }
}
```

مرتب سازی درجی

26, 5, 77, 1, 61, 11, 59, 15

مرتب سازی درجی

```
void insertion_sort(int list[], int n)
{
  int i, j,next;
 for (i=1; i<n; i++)
     {
      next= list[i];
      for (j=i-1; j>=0&& next<list[j];j--)
            list[j+1] = list[j];
      list[j+1] = next;
     }
}
```

مرتب سازی ادغامی

26, 5, 77, 1, 61, 11, 59, 15

مرتب سازی heap

26, 5, 77, 1, 61, 11, 59, 15

تحلیل زمانی روش های مرتب سازی