تقسیم و حل

مراحل اصلی تقسیم و حل

۱) تقسیم(Divide)

۲) حل یا غلبه(Conquer)

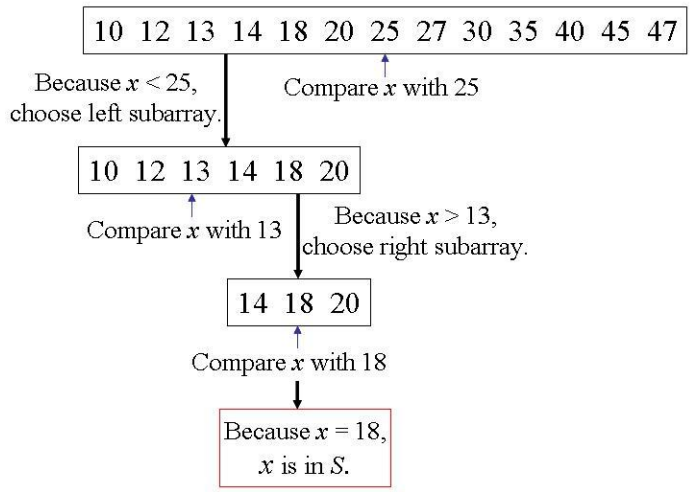
۳) ترکیب(Combine)

جستجوی دودویی (Binary Search)

مساله: تعیین اینکه عنصر x در آرایه مرتب شده S به طول n جود دارد؟

1. ***Divide*** the array into two subarrays about half as large. If x is smaller than the middle term, choose the left subarray; otherwise, choose the right subarray.
2. ***Conquer*** (solve) the subarray by determining whether x is in that subarray. Unless the subarray is sufficiently small, use recursion to do this.
3. *Obtain* the solution to the array from the solution to the subarray.

Problem: Determine whether x is in the sorted array S of size n.
Inputs: positive integer n, sorted array of keys S
indexed from 1 to n, and a key x.
Output: *location*, the location of x in S (0 if x is not in S)



شکل ۳-۱: روند جستجوی دودویی روی یک مثال

```
index location (index low, index high)
{
  index mid;
   if (low > high )
       return 0;
   else  {   mid = ⌊(low + high )/2⌋;
             if (x = = S[mid])
                location = mid ;
             else if  (x < S[mid])
                     return location (low, mid −1);
                  else return location (mid +1, high);      }
}
```

Alg. 3-1

Worst-Case Time Complexity

$$W(n) = W(\frac{n}{2}) + 1 \quad \text{and} \quad W(1) = 1$$

$$W(n) = \Theta(n^d \log_b n) = \Theta(\log_2 n)$$

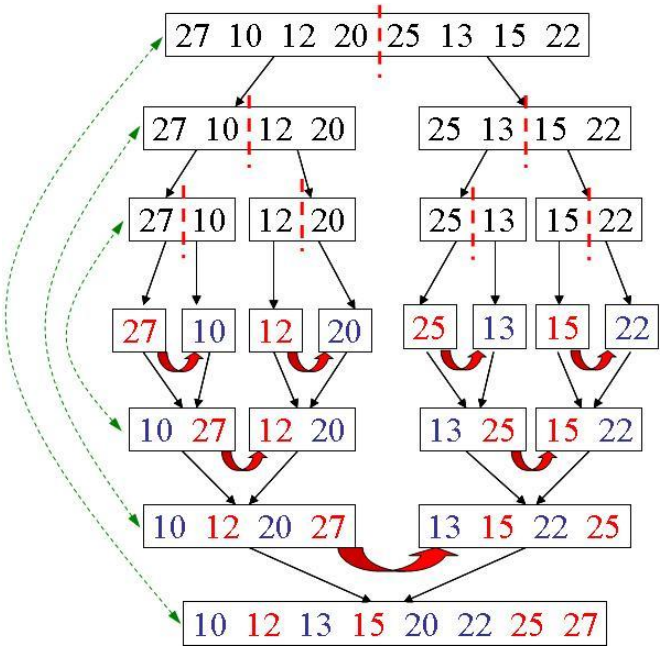
مرتب‌سازی ادغامی (Mergesort)

1. ***Divide*** the array into two subarrays each with n/2 items.
2. ***Conquer*** (solve) each subarray by sorting it recursively, unless the array is sufficiently small.
3. ***Combine*** the solutions to the subarrays by merging them into a single sorted array.

مثال:

27 10 12 20 25 13 15 22

1. Divide the array :
   27 10 12 20 and 25 13 15 22
2. Sort each array :
   10 12 20 27 and 13 15 22 25
3. Merge the subarrays :
   10 12 13 15 20 22 25 27



شکل ۳-۲: روند مرتب‌سازی ادغامی برای یک مثال

Problem: Sort n keys in non-decreasing sequence.
Inputs: positive integer n, array of keys S index from 1 to n.
Output: the array S containing the keys in non-decreasing sequence.

```
void mergesort (int n, keytype S[])
 { if ( n > 1)
      {
       const int  h = ⌊n/2⌋ , m = n − h ;
       keytype U[1..h] , V[1..m] ;
       copy  S[1] through S[h]  to U[1] through U[h];
       copy  S[h+1] through S[n]  to V[1] through V[m];
         mergesort (h, U);
         mergesort (m, V);
         merge (h, m, U, V, S);
      }
}
```

Alg. 3-2

```
void merge (int h, int m, const keytype U[], const
keytype V[], keytype S[])
 {
   index  i, j, k ;
    i = 1;  j = 1;  k = 1;
    while  ( i <= h && j <= m)
        {
          if  (U[i] < V[j])
              {
               S[k] = U[i];
                i++;
              }
           else{
                S[k] = V[j];
                    j++;
              }
            k++;
        }
    if (i > h)
    copy  V[j] through V[m]  to S[k] through S[h+m];
   else
     copy  U[i] through U[h]  to S[k] through S[h+m];
 }
```

Alg. 3-3

$W(n) = 2W(n/2) + (n/2+n/2-1) = 2W(n/2) + n-1$
$W(n) \in \Theta(n \log n)$

مثال: مراحل الگوریتم ادغام روبرو را روی دو لیست زیر نشان دهید.

k	U		V		S			
1	10	20	15	25				
2								
3								
4								

مرتب‌سازی سریع (Quicksort)

Also is called **Partition Exchange Sort**

Problem: Sort n keys in non-decreasing order.
Inputs: positive integer n, array of keys S index from 1 to n.
Outputs: the array S containing the keys in non-decreasing order.

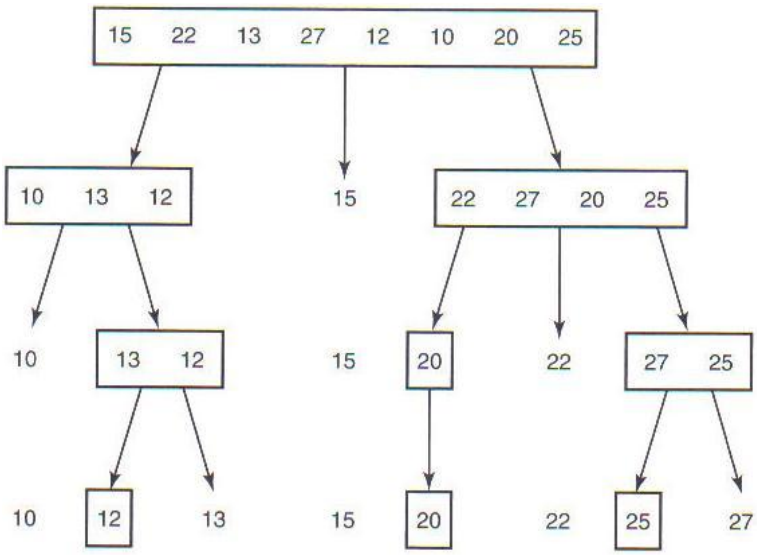
```
void quicksort (index low, index high)
{
  index   pivotpoint;

    if (high  > low)
      {
        partition (low, high, pivotpoint)
        quicksort ( low, pivotpoint −1) ;
        quicksort (pivotpoint+1, high)
      }
}
```

Alg. 3-4

مثال

Suppose the array : 15 22 13 27 12 10 20 25

1- Partition the array: all items < pivot are to the left and all items > pivot to the right
2- Sort the subarray.



شکل ۳-۳: روند کلی مرتب‌سازی سریع روی یک مثال

مثال: الگوریتم افراز روبرو را بر لیست زیر اعمال کنید.

i	j	S[1]	S[2]	S[3]	S[4]	S[5]
		15	22	13	27	12

```
void partition (index low, index high, index& pivotpoint)
{
    index i, j;
     keytype pivotitem;

     pivotitem = S[low];
     j = low;

    for (i = low + 1; i ≤ high; i ++)
          if (S[i] < pivotitem)
             {
                j ++;
                exchange S[i] and S[j];
             }
     pivotpoint = j;
     exchange S[low] and S[pivotpoint];
}
```

Alg. 3-5

آنالیز پیچیدگی الگوریتم مرتب‌سازی سریع

برای تحلیل الگوریتم باید نخست الگوریتم افراز (Partition) را تجزیه و تحلیل کنیم:

Every-Case Time Complexity (Partition)

Basic operation: the comparison of $S[i]$ with pivotitem
Input size:
$\boldsymbol{n = high - low + 1}$, the number of items in the subarray. Every item except the first is compared.
Hence,

$$T(n) = n - 1$$

تحلیل الگوریتم مرتب‌سازی سریع در بدترین حالت

Worst-Case Time Complexity (Quicksort)

$$\begin{aligned} W(n) &= W(0) + W(n-1) + n-1 \\ &= W(n-1) + n-1 \\ &= n(n-1)/2 \end{aligned}$$

$$\Rightarrow W(n) \in \Theta(n^2)$$

تحلیل الگوریتم مرتب‌سازی سریع در حالت میانگین

Average-Case Time Complexity(Quicksort)

Time to Partition

$$A(n) = \sum_{p=1}^{n} \frac{1}{n}\left[A(p-1) + A(n-p)\right] + (n-1)$$

Average time to sort
subarrays when pivotpoint is p

$$nA(n) = (n+1)A(n-1) + 2(n-1)$$

$$\therefore \frac{A(n)}{n+1} = \frac{A(n-1)}{n} + \frac{2(n-1)}{n(n+1)} \Rightarrow a_n = a_{n-1} + \frac{2(n-1)}{n(n+1)}$$

$$A(n) \approx 2(n+1)\ln n \in \Theta(n \log n)$$

$$P.S.\ \sum_{i=1}^{n} \frac{1}{i} \approx \ln n \quad \text{for large } n \text{ (area under } \frac{1}{x}\text{)}$$

تمرین‌ها

تمرین۳-۱: بررسی کنید چگونه اثبات می‌شود که الگوریتم مرتب‌سازی سریع در حالت میانگین از $\Theta(n \log n)$ برخوردار است.