روش‌های پایین به بالا - جدول بازگشتی دینامیک

جستجوی خطی

در برنامه‌های کاربردی استفاده می‌شود و روش جستجو به این ترتیب است که اندیس عنصر مورد جستجو (کلید) با عنصر اول مقایسه می‌شود در صورت عدم تساوی با عنصر دوم مقایسه می‌شود

بدترین حالت: عنصر مورد جستجو آخرین عنصر آرایه باشد. n مقایسه
بهترین حالت: عنصر مورد جستجو اولین عنصر آرایه باشد. ۱ مقایسه
در حالت متوسط: $\frac{n+1}{2}$ مقایسه

مرتبه اجرایی O(n)

```
int SeqSearch (items S[n], items x)
{
    int location=1;
    while(location<=n
                        S[location]!=x)
        Location = location+1;
    if (location>n)
        return 0;
    else
        return location;
}
```

مثال: در آرایه زیر عدد 43 را جستجو کنید به روش جستجوی ترتیبی.

1	2	3	4	5	6	7	8	9
21	7	39	41	15	43	10	9	11

برای پیدا کردن 43 نیاز است 6 مقایسه صورت گیرد اگر 43 آخرین عنصر بود نیاز است 9 مقایسه و اگر اولین عنصر باشد یک مقایسه نیاز است برای همین از این روش جستجو کمتر استفاده می‌کنیم

$O(\log n)$ وقت التشغيل: الحل

24 دقيقة

```
int bsearch(int x[], int n, int m)
{
    int low, high, mid;
    low = 0; high = n-1;
    while (low <= high)
    {
        mid = ((low+high)/2;
        if (m < x[mid])
            high = mid-1;
        else if (m > x[mid])
            low = mid+1;
        else
            return mid;
    }
    return -1;
}
```

1	2	3	4	5	6	7
8	12	16	20	22	24	31

low ↑ mid ↑ high

mid = $\frac{1+7}{2}$ = 4

x[4] = 20

m > x[mid]

2 > x[4]

low = mid + 1 = 4 + 1 = 5

high = 7

mid = $\frac{7+5}{2}$ ، 6 m = x[6]

جلسه هفتم حل اعی آلگوریتم صفحه ۱

ضروری: تقسیم و غلبه

در این تکنیک مسئله به دو یا چند نمونه کوچکتر تقسیم می‌شود. سپس این نمونه‌های کوچکتر حل شده و با ترکیب حل این نمونه‌های کوچکتر، مسئله اصلی حل می‌شود.

Top - Down روش حل

تحلیل پیچیدگی زمانی همتجوی دودویی

بدترین حالت: عدد صدد جستجو (x) آخرین عضو آرایه باشد یا آرایه فرد باشد یا اصلا عضو نباشد.

$$w(n) = w\left(\frac{n}{2}\right) + 1$$

$$w(1) = 1$$

نکته

$$T(n) = a\,T\left(\frac{n}{b}\right) + cn^k \Rightarrow \begin{cases} \theta(n^{\log_b^a}) & a > b^k \text{ اگر} \\ \theta(n^k \log_2^n) & a = b^k \\ \theta(n^k) & a < b^k \end{cases}$$

در همتجوی دودویی $k=0,\ b=2,\ a=1$

$$a = b^k$$

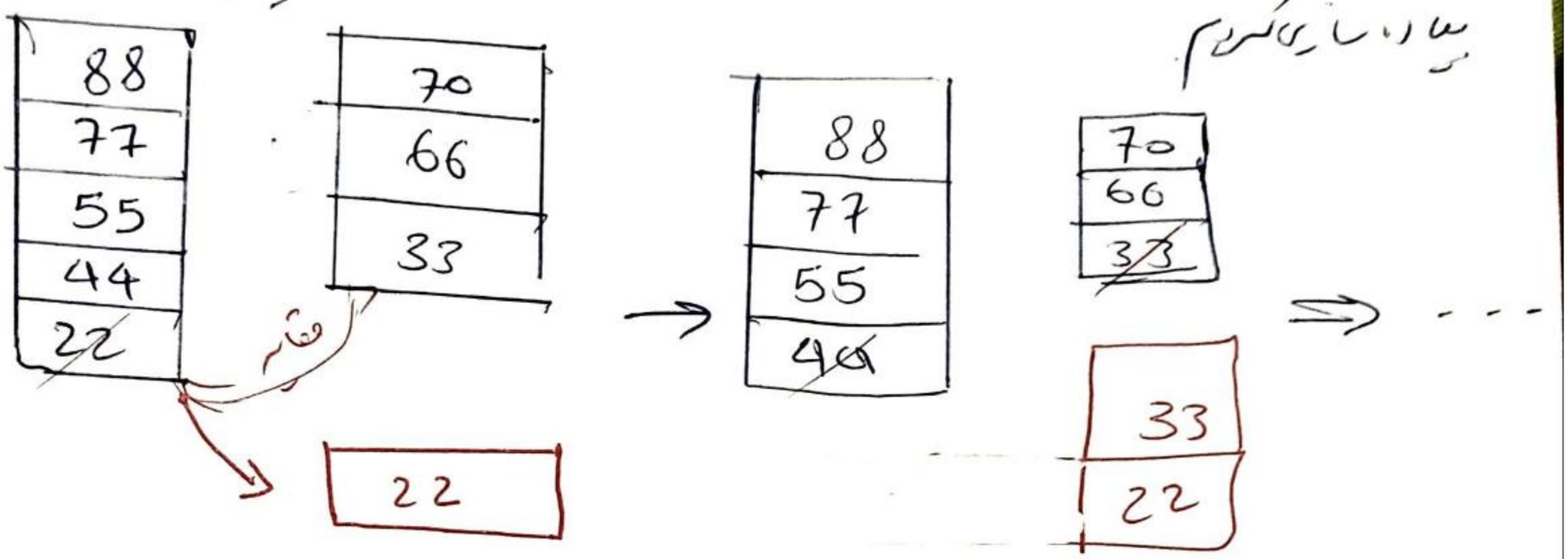
$$1 = 2^0$$

پس $a=b^k$ برقرار است پس

$$\theta(n^k \log_2^n) \Rightarrow \theta(n^0 \log_2^n) \Rightarrow \theta(\log_2^n)$$

مرتب سازی ادغامی

۲ دسته کارت مرتب شده داریم در هر مرحله کارت حلویی هر دسته را با کارت حلویی دسته دیگر مقایسه میکنیم و کارت با شماره کوچکتر را در دسته جدید قرار می دهیم من در ادامه آنرا پیاده سازی کردم



22, 33, 44, 55, 66, 70, 77, 88

تابع مرتب سازی ادغامی که ۳ آرایه A, B, S را دارد و تعداد A، p و تعداد B، m است

```
Vaid merge (int p, int m, int A[], int B[], int S[])
  { int i, j, k;
      i = j = k = 1;
    while ( i <= p && j <= m ) {
      if (A[i] < B[j]) {
        S[k] = [ ];
      }      i++;
      else  S[k] = B[j];
    k++;
  }
  if (i > p)
    while (j <= m)
    { S[k] = B[j];
      k++;
    } j++;
```

```
else
  if (j > m)
while (i <= p)
  {
  S[k] = A[i];
    k++;
  } i++;
```

بدترین حالت در مرتب سازی ادغامی چه زمانی رخ می دهد؟

همزمان هر آرایه باطول کمتر از اندیس عنصر آرایه
باطول بیشتر کوچکتر باشد.

مثلاً در شکل زیر به ۴ مقایسه نیاز داریم

$$B(p,m) = \min(p,m)$$

بدترین حالت زمانی رخ می دهد که تمامی عناصر آرایه اول (یا جز آخرین عضو آن) از کوچکترین عنصر آرایه دوم کوچکتر باشد ولی این آخرین عنصر از تمام عناصر آرایه دوم بزرگتر باشد.

$$w(p,m) = p+m-1$$

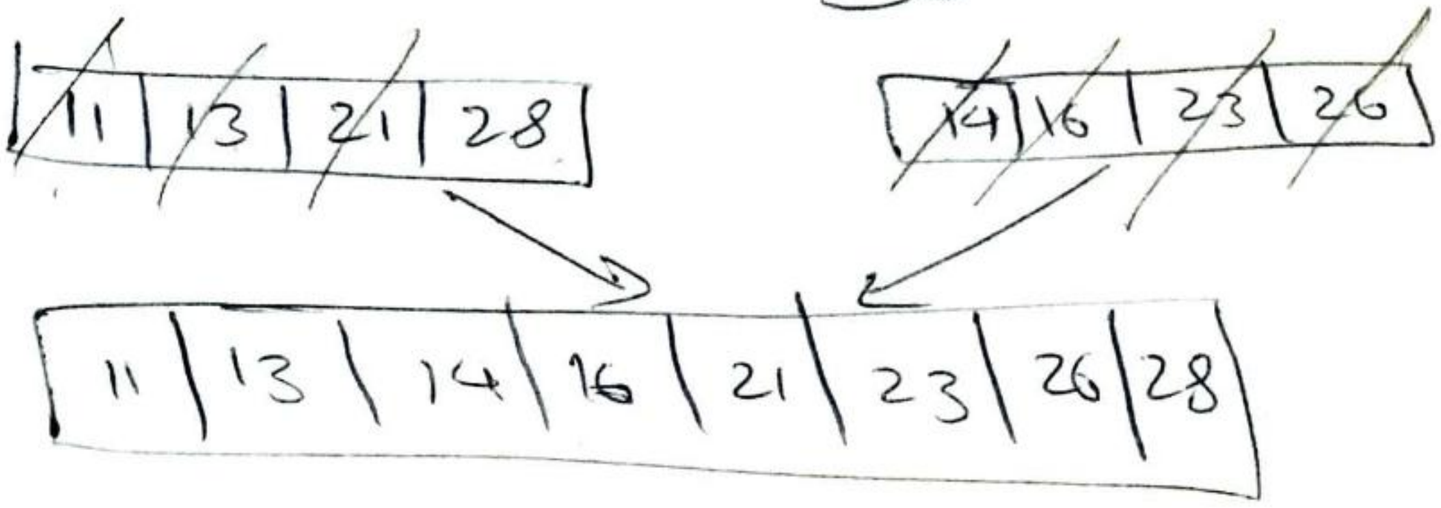
$$4+5-1=8$$

Merge Sort

مرتب سازی ادغامی

① تقسیم آرایه به دو نیمه

② با روش ادغام هر کدام از آرایه ها مرتب شده را به صورت صعودی ادغام می کنیم.



تابع مرتب سازی ادغامی

```
Void mergesort(int n, int S[7])
{
  int p=⌊n/2⌋ , m=n-p;
  int A[1..p] ,B[1..m]
  if (n>1) {
   Copy S[1] through S[p] to A[1] through A[p]
   Copy S[p+1] through S[n] to B[1] through B[m];
   mergesort (p,A);
    mergesort (m,B);
  } } Merge(p,m,A,B,S);
```

تحلیل پیچیدگی

$$\omega(n) = \omega(p) + \omega(m) + (p+m-1)$$

$\omega(p)$: زمان لازم برای مرتب سازی آرایه A

$\omega(m)$: زمان لازم برای مرتب سازی آرایه B

$(p+m-1)$: زمان لازم برای ادغام A, B

$$p = \lfloor n/2 \rfloor = \frac{n}{2} \qquad m = n-p = n-\frac{n}{2} = \frac{n}{2}$$

$$\omega(n) = \omega\left(\frac{n}{2}\right) + \omega\left(\frac{n}{2}\right) + \left(\frac{n}{2}+\frac{n}{2}-1\right)$$

$$\omega(n) = 2\omega\left(\frac{n}{2}\right) + n - 1$$

فرم $aT\left(\frac{n}{b}\right) + cn^k$

$a=2$, $b=2$, $K=1$ $\Rightarrow$ $a = b^K$ ($2 = 2^1$)

$$\theta(n^1 \log_2^n)$$

Quick Sort

مرتب‌سازی سریع

در این روش در هر گام یک عنصر محوری یا (Pivot) گیر را عنصر به مکان اصلی

مثال ← 26 5 37 1 61 11 59 15 48 19

با بررسی Quicksort مرتب کنید.

در ابتدای ۲ اشاره‌گر داریم left و right به ابتدا و انتهای آرایه

اشاره‌ی کم عنصر گوه را 26 در نظر می‌گیریم.

i = left;
j = right+1;

left = 1 ، i = 1 نام‌ترین اعضای شروع به حرکت

right = 10 ، j = 11 بزرگ‌تر از گوه (یعنی 26) برسند.

از سمت راست هم آنقدر می‌رود تا به عنوانی کوچک‌تر از 26 برسد و

جای این دو عدد با هم عوض می‌شود

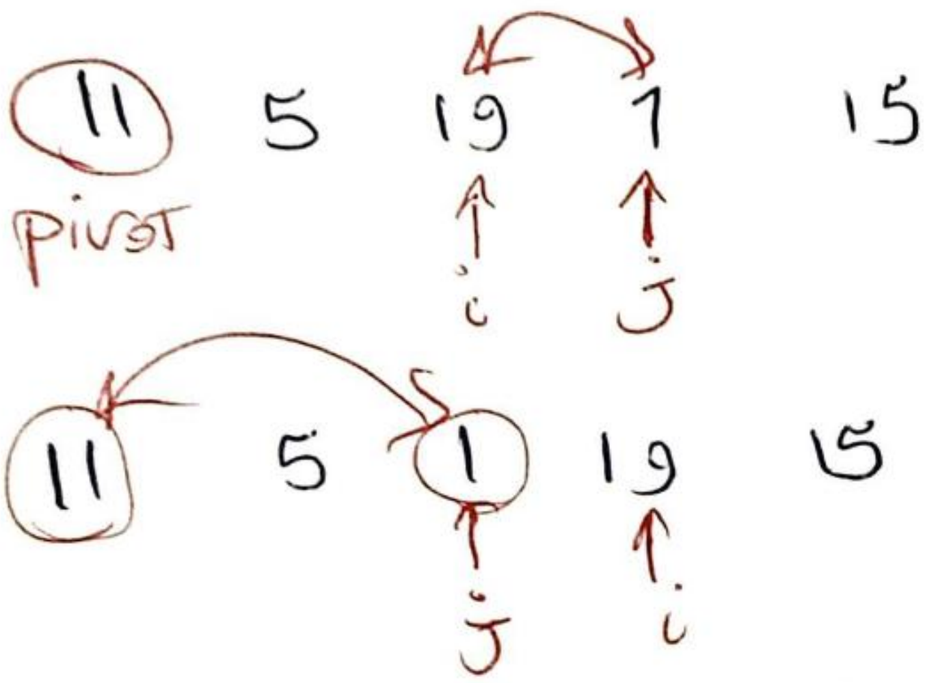
گام‌های از هم رسیدند باید pivot را با خانه‌ای سرخ به آن اشاره می‌کند عوض کنیم

یعنی 26 با 11

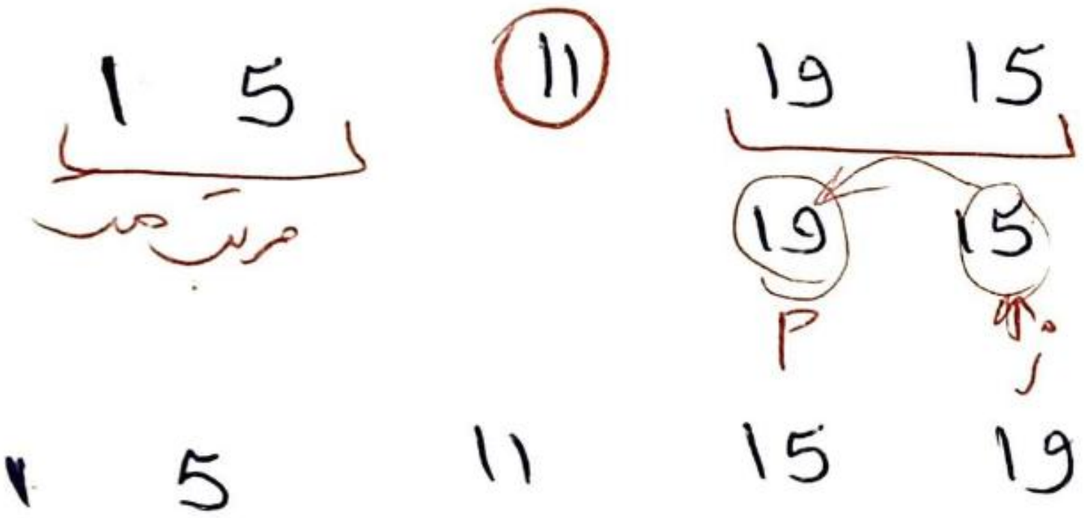
مثال

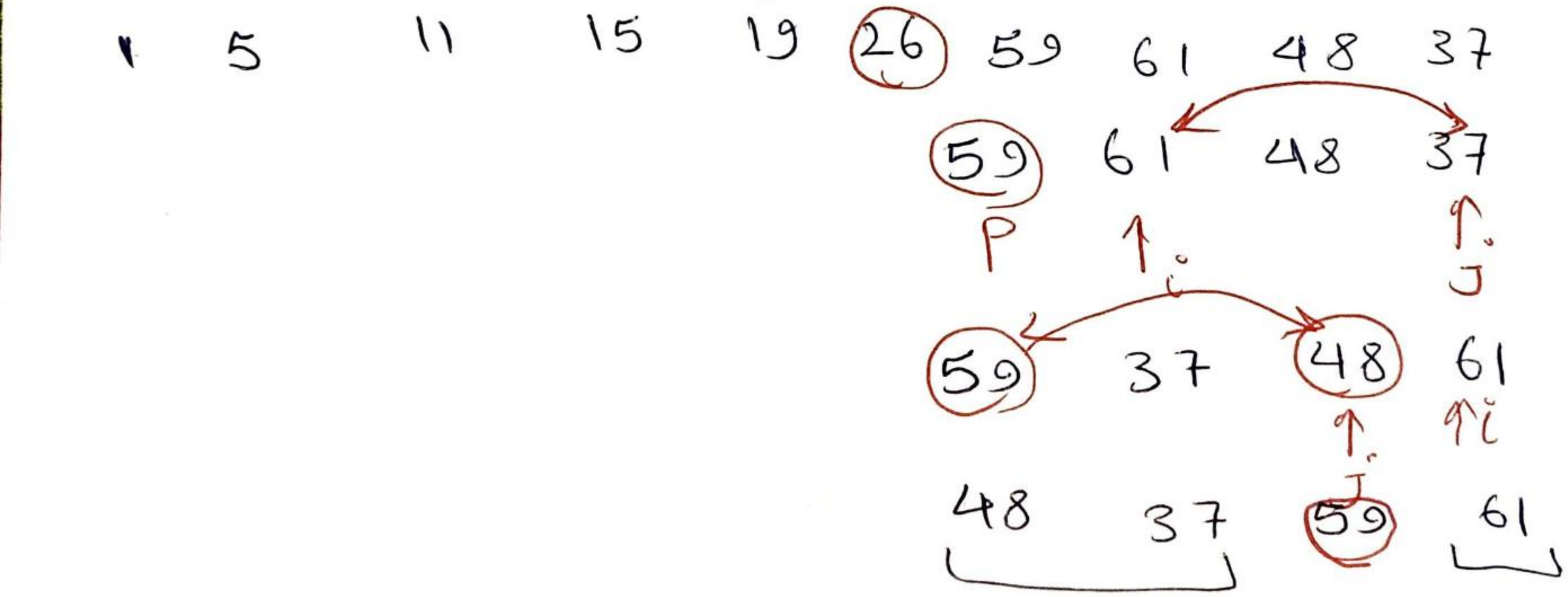
11 5 19 1 15 26 59 61 48 37

مرتب سازی این



از هم رد شدند پس pivot با عضو غیر مرتب عوض میشه





37 48 59 61

1 5 11 15 19 26 37 48 59 61

الگوریتم مرتب سازی سریع

```
procedure partition(int left,right, pivotpoint; items x)
{
int i, j, pivot ;
    i=left ; j=right+1 ;  pivot = x[left];
repeat
    repeat
        i = i+1;
until  x[i]>= pivot;
 repeat
        j = j -1 ;
until x[j]<= pivot;
    if (i<j)
      swap (x[i], x[j]);
until  i>=j;
 swap (x[left,x[j]);
 Pivotpoint=j;
}
procedure @uicksort(items x, int left, right)
{int  pivotpoint;
} if( left < right)
      { partition (x, left, pivotpoint-1)
        @uick sort(x, left, pivotpoint-1)
        @uicksort(x, pivotpoint+1, right)
  }
}
```

جزا بزرگی عنصر بولا بردل دلقی قرار

مرتب سازی زیر لیست چپ

مرتب سازی زیر لیست راست

تحلیل پیچیدگی

بدترین حالت در مرتب سازی سریع زمانی اتفاق می افتد که آرایه از قبل مرتب باشد چرا که تابع partition در هر بار مقدار left را به عنوان pivot او pivot در نظر میگیرد بنابراین آرایه به طور مکرر به صورت یکی از آرایه در سمت چپ تهی و دیگری آرایه با n-1 عنصر در سمت راست قرار می گیرد.

مقایسه

$$\omega(n) = \omega(0) + \omega(n-1) + n-1$$

$$\omega(n) = \omega(n-1) + n-1$$

$$\omega(1) = \omega(0) + 0$$

$$\omega(1) = 0$$

$$\omega(2) = \omega(1) + 1 = 1$$

$$\omega(3) = \omega(2) + 2 = 3$$

$$\omega(4) = \omega(3) + 3 = 6$$

$$\omega(5) = \omega(4) + 4 = 10$$

$$\omega(n) = \frac{n(n-1)}{2}$$

$$\omega(n) = \frac{n^2 - n}{2}$$

$$\theta(n) = n^2$$