جستجوی خطی در آرایه

```
Array A[n] , x
Int search (A[n] , x) ;
{
    int i = 1 ;
    while (i <= n && A[i] != x)
        i + + ;
    if (i > n ) return – 1 // داده پیدا نشده
    else return i // داده در محل اندیس آرایه است
}
```

1	2	3	4	5
1	5	2	8	6

n	x	i	A[i]
5	8	1	1
		2	5
		3	2
		4	8

x = A[4]

جستجوی دودویی برای آرایه‌های مرتب

```
Int bsearch (A[n] , int x , int L , int U)
{
    int i ;
    while
    {
        i = ⌈(L + U) / 2⌉ ;
        if ( x < A[i] ) U = i – 1
        else if ( x > A[i] ) L = i + 1
            else return i // داده در اندیس i است
    }
    return – 1 // داده پیدا نشده است
}
```

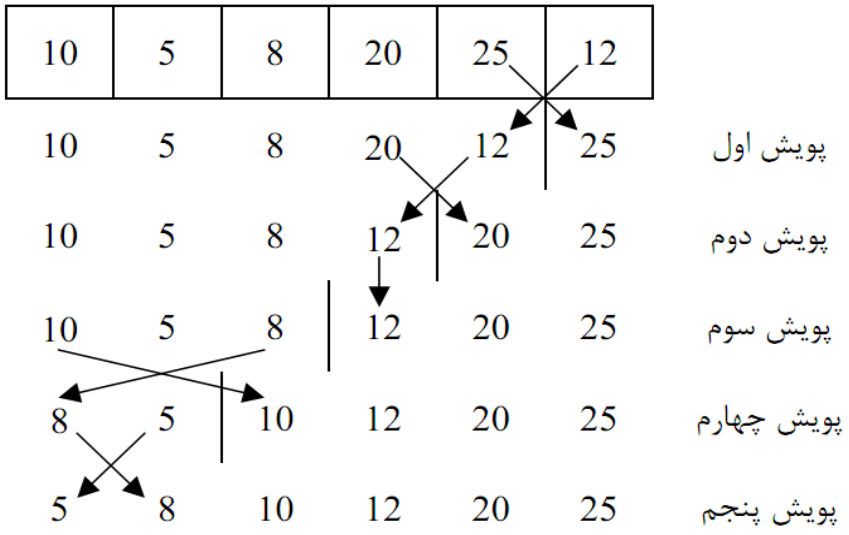
مرتب‌سازی

در مرتب‌سازی تعدادی عنصر که از ورودی داده شده‌اند را بر اساس کلیدشان بصورت صعودی یا نزولی مرتب می‌کنیم.

مرتب‌سازی انتخابی (Selection Sort)

در مرتب‌سازی انتخابی یک آرایه n عنصری $(A[1..n])$, $n-1$ بار پیمایش می‌شود. در هر پیمایش بزرگتـرین عنصر در محل درست خود یعنی انتهای آرایه قرار می‌گیرد. با این روش آرایه از انتها مرتب می‌شود. در مرتب‌سازی انتخابی می‌توان با انتخاب کوچکترین عنصر در هر پیمایش و قرار دادن آن در محل درست خود یعنی ابتدای آرایه در هر پیمایش , مرتب‌سازی را از ابتدای لیست انجام داد.

مثال :



برنامه کلی مرتب‌سازی انتخابی به شرح ذیل می‌باشد :

```
for (i = n ; i > 1 ; – – 1)
{
        max = A[1] ;
        index = 1 ;
        for (i = 2 ; j < = i ; + + j)
        if (A[j] > max)
        {
                max = A[j] ;
                index = j ;
        }
        A[index] = A[i] ;
        A[i] = max ;
}
```

مثال :

3	2	5	1
1	2	3	4

i	j	n	max	index
4	2	4	3	1
3	4		5	3
	4		3	1
	2			
	3			

نکـــته : در مـرتب‌سازی انتخابی , حداکثر و حداقل n^2 مقایسه داریم. حداقل جابجایی صفر و حداکثر جابجایی نیز n بار خواهد بود.

مرتب‌سازی حبابی (Bubble Sort)

در مرتب‌سازی حبابی یک آرایه n عنصری (A[1..n]) , n − 1 بار پیمایش می‌شود و در هر پیمایش دو عنصر متوالی با یکدیگر مقایسه شده که در صورت لزوم جابجا خواهند شد. در هر پیمایش , طول آرایه پیمایش شده نسبت به مرحله قبل یکی کم می‌شود.

10	30	50	40	90	20	
10	30	40	50	20	90	مرحله اول
10	30	40	20	50	90	مرحله دوم
10	30	20	40	50	90	مرحله سوم
10	20	30	40	50	90	مرحله چهارم
10	20	30	40	50	90	مرحله پنجم

مرتب‌سازی از انتهای لیست

```
for (i = 1 ; i < n ; + + i)
        for (j = 1 ; j < = n ; + + j)
        if (A[j]) > A[j + 1])
        swap (A[j] , A[j + 1]) ;
```

مرتب‌سازی از ابتدای لیست

```
for (i = 1 ; i < n ; + + i)
        for (j = n ; j > = i ; – – j)
        if (A[j]) < A[j – 1])
        swap (A[j] , A[j – 1]) ;
```

مثال :

5	3	7	2	
1	2	3	4	
2	5	3	7	پویش اول
2	3	5	7	پویش دوم

i	j	n
1	4	4
2	3	
3	2	
	1	
	4	

« الگوریتم متعادل است »

در هر پویش امکان n^2 جابجایی وجود دارد. حداقل تعداد جابجایی نیز صفر است.

مرتب‌سازی حبابی بهینه شده

```
for (i = 1 ; i < = x ; + + i)
{
        sw = 0 ;
        for (j = 1 ; j < n – 1 ; + + j)
        if (A[j] > A[j + 1])
        {
                sw = 1 ;
                swap (A[j] , A[j + 1]) ;
        }
        if (sw = = 0) break ;
}
```

مرتب‌سازی دَرجی (Insertion Sort)

در مرتب‌سازی درجی فرض شده است که $i - 1$ عنصر اول لیست مرتب هستند. پس عنصر iام در جای صحیح خود قرار دارد.

```
For (i = 2 ; i < = n ; + + i)
{
        y = A[i] ;
        j = i – 1 ;
        while (j > 0 & & (y < A[j]))
        {
                A[j + 1] = A[j] ;
                j = j – 1 ;
        }
        A[j + 1] = y ;
}
```

50	60	40	20	10	30
1	2	3	4	5	6

50	60	40			

50	40	60			

40	50	60			

10	40	50	60		

i	n	y	j
2	6	60	1
3		40	2
4		20	1
5		10	0
			3
			2
			1
			0
			4
			3
			2
			1
			0

در این جابجایی حداقل n تا پویش داریم و در بهترین حالت نیز جابجایی نداریم.

مثال : آرایه زیر را به روش درجی مرتب کنید.

	1	2	3	4	5
A	4	8	5	2	6
	4	8			
	4	5	8		
	2	4	5	8	
	2	4	5	6	8

n	i	j	y
5	2	1	8
	3	2	5
	4	1	2
		3	6
		2	
		1	
		0	
		4	
		3	