طراحی الگوریتم

موسسه آموزش عالی پاسارگاد شیراز

روش تقسیم و غلبه

- در این روش برای حل یک مسئله آن مسئله را به مسائل کوچکتر تقسیم می کنیم و سپس شروع به حل مسائل کوچکتر می کنیم. در صورتی که قادر به حل مسئله کوچکتر نبودیم آن را نیز به مسائل کوچکتر تقسیم می کنیم و این کار را تا زمانی ادامه می دهیم که مسئله کوچک به دست آمده قابل حل باشد. پس از حل مسئله کوچکتر،نتیجه مسئله را به عنوان جواب برای مسائل بزرگتر استفاده می کنیم.
- در واقع این روش،روش بالا به پایین (top down) است.

روش تقسیم و غلبه

- یکی از مسائل مهم که به روش تقسیم و غلبه حل می شود مسئله جستجوی دودویی می باشد که در آن هر بار مسئله ی بزرگتر به یک مسئله با بزرگی نصف مسئله اصلی تبدیل می شود.

```
int b_search(int list[], int l,int h, int x)
{
        if (l > h)
                return -1;
        int m = (l + h) / 2;
        if (list[m] == x)
                return m;
        if (list[m] > x)
                return b_search(list, l, m - 1, x);
        if (list[m] < x)
                return b_search(list, m+1, h, x);
}
```

روش تقسیم و غلبه

- یکی از مسائل مهم که به روش تقسیم و غلبه حل می شود مسئله جستجوی دودویی می باشد که در آن هر بار مسئله ی بزرگتر به یک مسئله با بزرگی نصف مسئله اصلی تبدیل می شود.

```
int b_search(int list[], int l,int h, int x)
{
        if (l > h)
                return -1;
        int m = (l + h) / 2;
        if (list[m] == x)
                return m;
        if (list[m] > x)
                return b_search(list, l, m - 1, x);
        if (list[m] < x)
                return b_search(list, m+1, h, x);
}
```

$$T(n)=\begin{cases}1 & \text{عنصر وسط آرایه}\\ T(\frac{n}{2})+1 & else\end{cases}$$

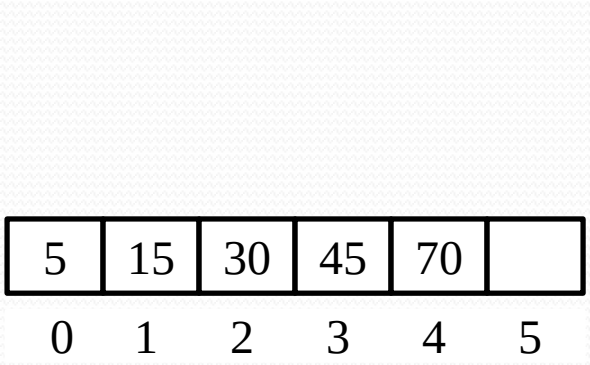
a=1

b=2

$$\theta(log_2^n)$$

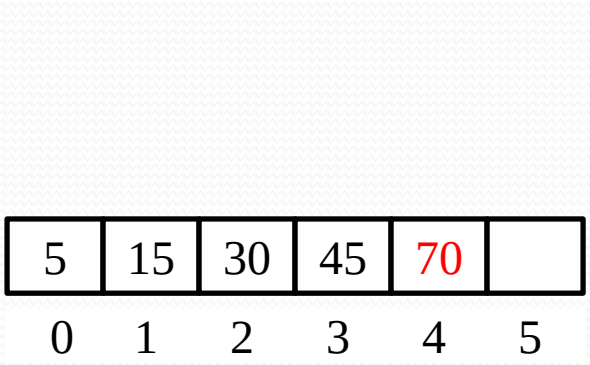
الگوریتم های مرتب سازی

- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم



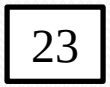
الگوریتم های مرتب سازی

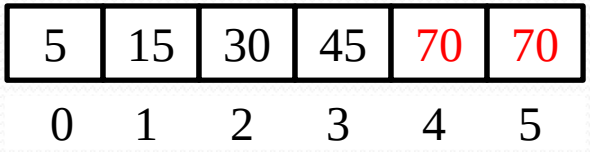
- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم



الگوریتم های مرتب سازی

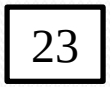
- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم

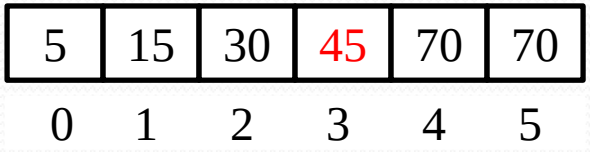




الگوریتم های مرتب سازی

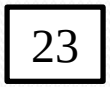
- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم

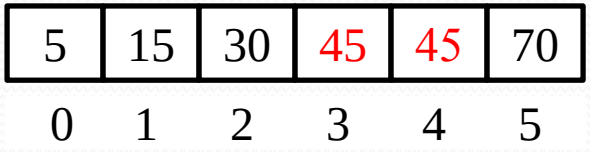




الگوریتم های مرتب سازی

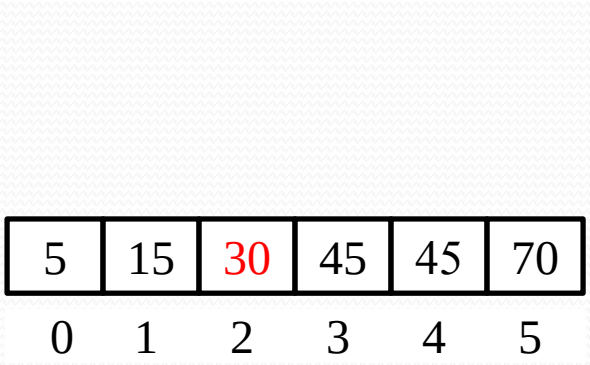
- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم





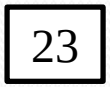
الگوریتم های مرتب سازی

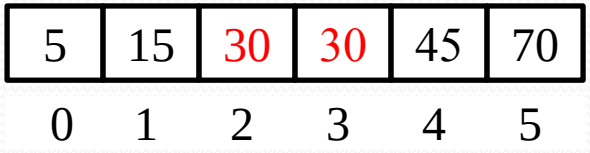
- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم



الگوریتم های مرتب سازی

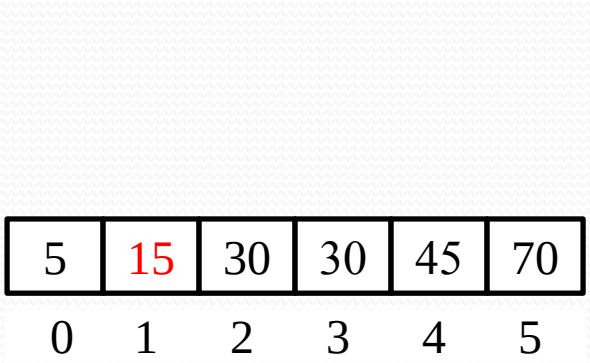
- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم





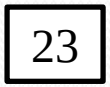
الگوریتم های مرتب سازی

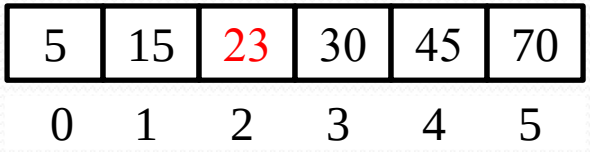
- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم



الگوریتم های مرتب سازی

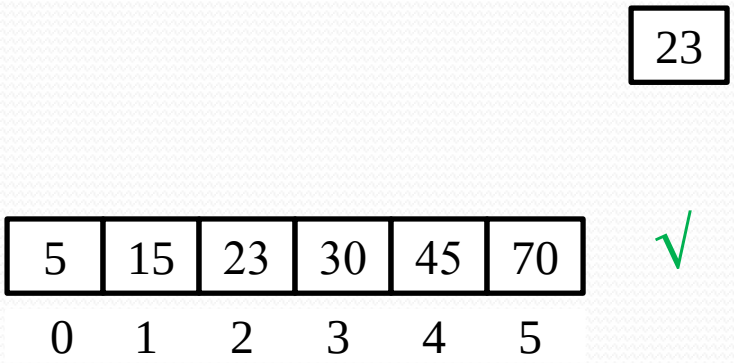
- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم





الگوریتم های مرتب سازی

- مرتب سازی درجی (insertion sort)
- در این الگوریتم در هر لحظه عملیات درج یک عنصر را در یک آرایه مرتب انجام می دهیم
- به عنوان مثال فرض کنید در آرایه مرتب زیر بخواهیم عمیات درج عدد ۲۳ را در آرایه انجام دهیم



الگوریتم های مرتب سازی

- مرتب سازی درجی (insertion sort)

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

مکان مناسب برای عنصر k

الگوریتم های مرتب سازی

- مرتبه اجرایی مرتب سازی درجی (insertion sort)
- بهترین حالت زمانی اتفاق می افتد که آرایه از قبل مرتب باشد در این حالت هیچ گونه عملیات جابجایی صورت نمی گیرد در نتیجه مرتبه اجرایی برابر با $o(n)$ می باشد.
- بدترین حالت زمانی اتفاق می افتد که آرایه به صورت برعکس مرتب باشد که در این حالت تقریبا هر بار باید به اندازه i بار عملیات shift انجام شود که در این حالت مرتبه اجرایی $1+2+3+4\ldots+n=\frac{n(n+1)}{2}\rightarrow O(n^2)$
- در حالت متوسط نیز مرتبه اجرایی $O(n^2)$ می شود

الگوریتم های مرتب سازی

- مرتبه اجرایی مرتب سازی درجی (insertion sort)

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

بهترین حالت O(n)

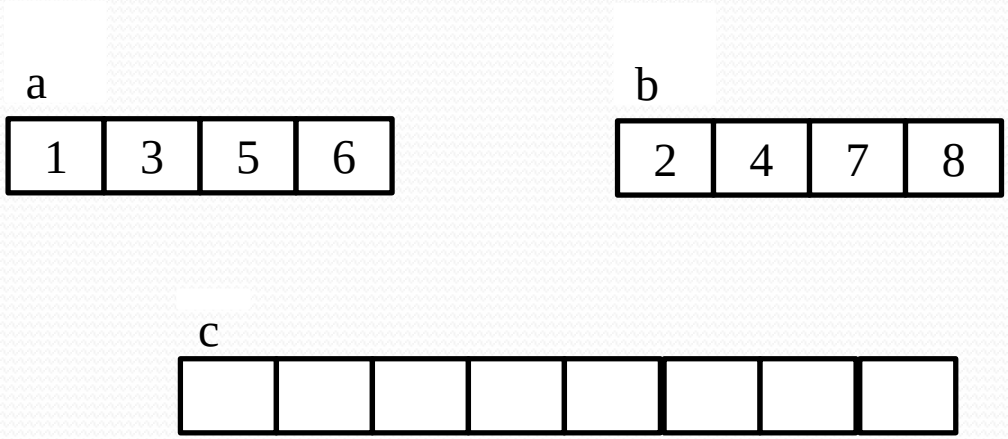
حالت متوسط و بدترین حالت $O(n^2)$

الگوریتم های مرتب سازی

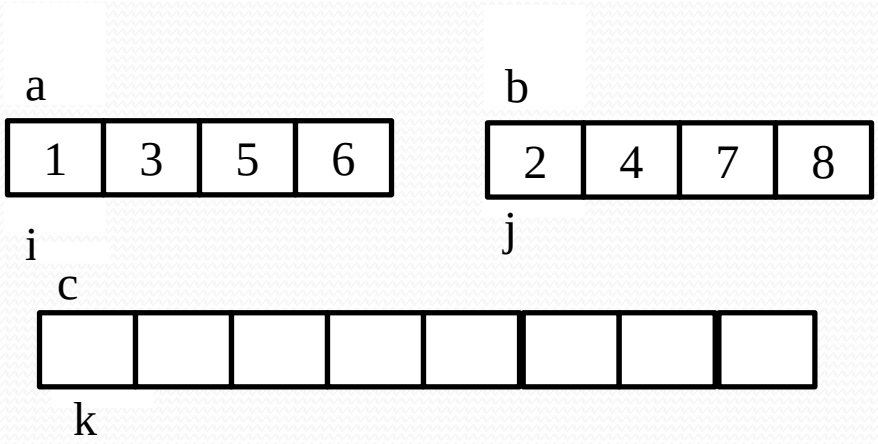
- الگوریتم مرتب سازی ادغامی(merge sort)
- برای مرتب سازی ادغامی، ابتدا الگوریتم مربوط به ادغام دو لیست مرتب را انجام می دهیم.

الگوریتم های مرتب سازی

- الگوریتم مرتب سازی ادغامی(merge sort)
- برای مرتب سازی ادغامی، ابتدا الگوریتم مربوط به ادغام دو لیست مرتب را انجام می دهیم.



الگوریتم های مرتب سازی



برای افزودن عنصر به آرایه c ممکن است سه حالت اتفاق بیافتد

۱)اگر a[i]>b[j]باشد

مقدار b[j]را در c[k]ذخیره می کنیم

و اندیس j و k را یک واحد افزایش می دهیم.

۲)اگر a[i]<b[j]باشد

a[i] را در c[k] ذخیره می کنیم

و اندیس i و k را یک واحد افزایش می دهیم.

۳)اگر a[i]==b[j]باشد

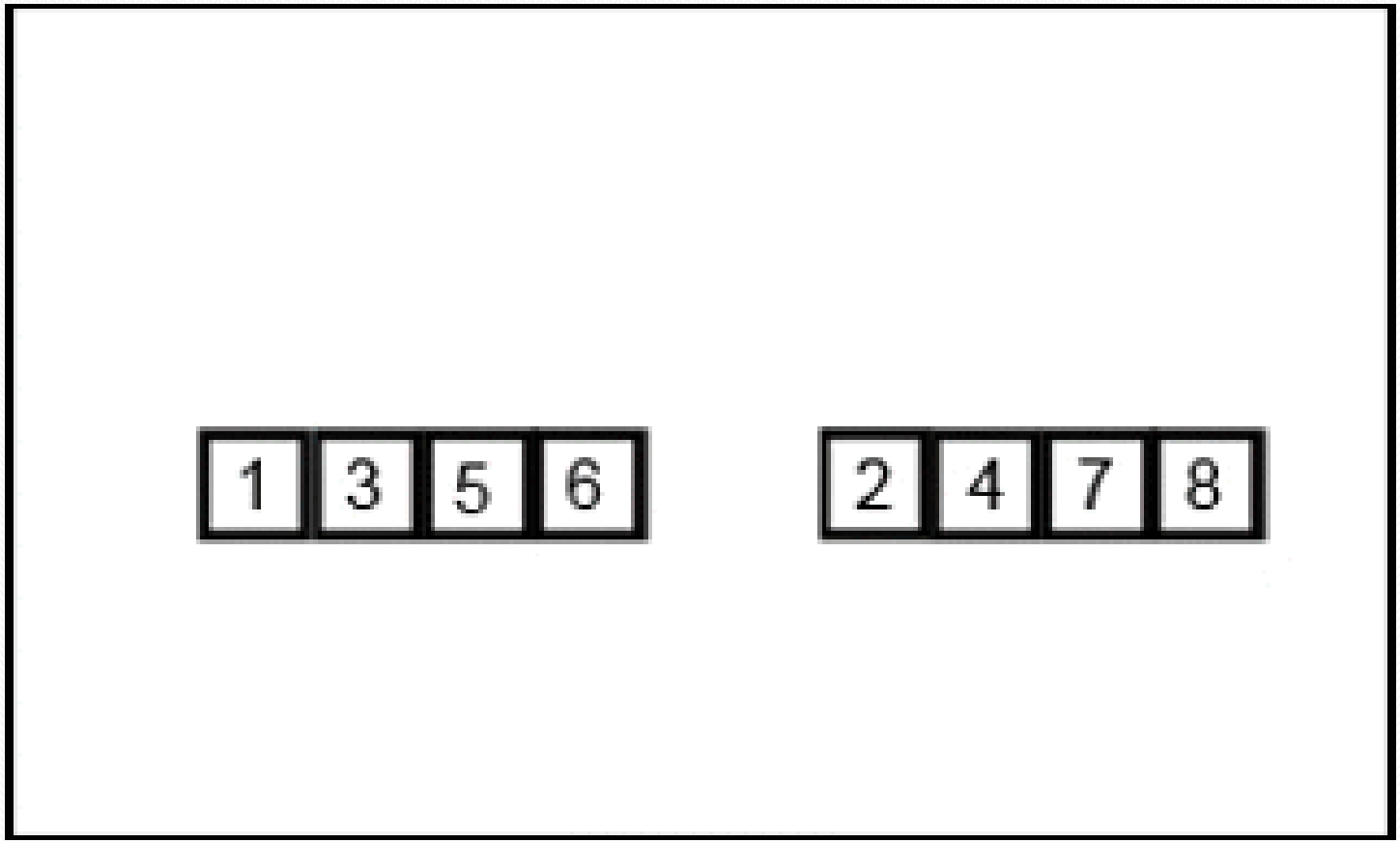
یکی را در c[k] ذخیره می کنیم

و اندیس های i و j و k را یک واحد افزایش می دهیم.

نکته : اگر به انتها یکی از لیست ها رسیدیم، مقادیر لیست دیگر را در انتها لیست c ذخیره می کنیم.

الگوریتم های مرتب سازی

- ادغام دو لیست مرتب

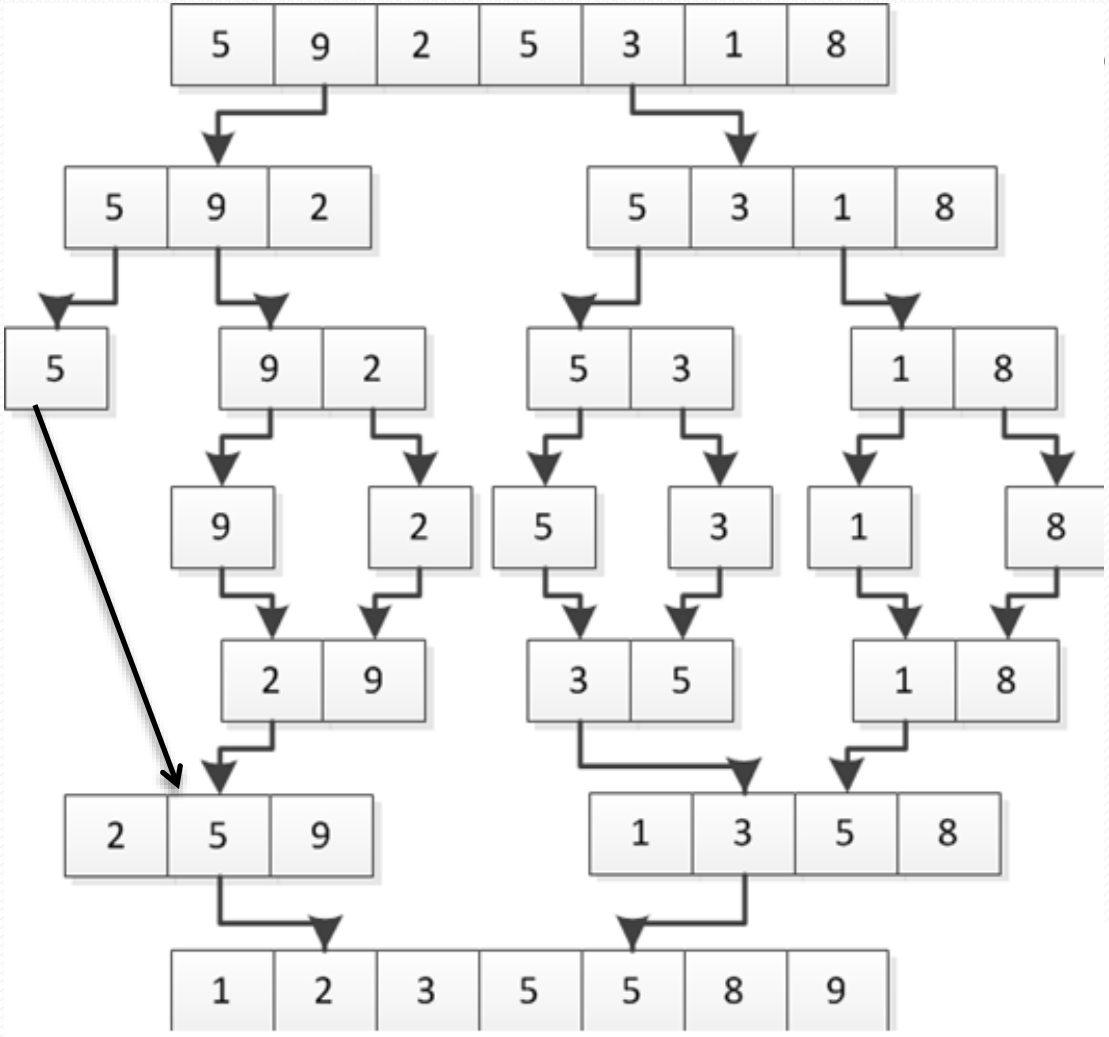


ادغام دو لیست مرتب

```
void merge_list( int a[], int b[], int c[], int m, int n )
{
        int i=0, j=0, k=0;
        while (i<m && j<n)
        {
                if (a[i]<b[j])
                {
                        c[k] = a[i]; i++;
                }else
                {
                        c[k] = b[j]; j++;
                }
                k++;
        }
        while (j<n)
        {
                c[k] = b[j];
                j++;k++;
        }
        while (i<m)
        {
                c[k] = a[i];
                i++;k++;
        }
}
```

الگوریتم های مرتب سازی

- الگوریتم مرتب سازی ادغامی



انجام عملیات ادغام

الگوریتم های مرتب سازی

- الگوریتم مرتب سازی ادغامی(merge sort)

```
void merge_sort(int s[], int n)
{
        int p = n / 2;
        int m = n - p;
        if (n>1)
        {
                int *a;
                int *b;
                int i;
                a = new int[p];
                for (i=0; i<p; i++)
                        a[i] = s[i];
                b = new int[m];
                for (int j = 0; j<m; j++, i++)
                        b[j] = s[i];
                merge_sort(a,p);
                merge_sort(b,m);
                merge_list( a, b, s,p,m);
        }
}
```

الگوریتم های مرتب سازی

- مرتبه اجرایی merge sort
- ابتدا مرتبه اجرایی merge list را محاسبه می کنیم
  - قسمت اصلی مقایسه عنصر دو آرایه است. بهترین حالت زمانی اتفاق می افتد که تمام عناصر لیست کوچکتر از کوچکترین عضو لیست بزرگتر،کوچکتر باشد.که در این بهترین حالت o(min(n , m))

a

15	18	30	48	50

b

2	7	9	11

- بدترین حالت که مجبوریم تمام مقایسه ها را انحام دهیم o(n+m-1)

a

8	15	18	20	42

b

2	12	34	50

الگوریتم های مرتب سازی

- مرتبه اجرایی merge sort با کمک merge list
  - دو بار فراخوانی merge sort هر بار با اندازه n/2
  - یک بار merge list با مرتبه O(n)

$$T(n)=\begin{cases}1 & n==1\\ 2T(\frac{n}{2})+O(n) & else\end{cases}$$

a=2
b=2
$\theta(f(n))=O(n)$

$$\text{مرتبه اجریی}=\begin{cases}\theta\left(n^{log_2^2}\right) & \theta\left(n^{log_2^2}\right)>n\\ n & \theta\left(n^{log_2^2}\right)<n\\ \theta(n*\log n) & \theta\left(n^{log_2^2}\right)=n \quad \surd\end{cases}$$

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort
- در این روش ابتدا یک عنصر را انتخاب می کنیم که به آن عنصر لولا(pivot) می گوییم سپس مکان مناسب عنصر لولا در آرایه را پیدا کرده و عنصر لولا را درآن مکان قرار می دهیم مکان مناسب عنصر لولا باید دارای خصوصیات زیر باشد
  - ۱-اگر مرتب سازی به صورت صعودی باشد عنصر لولا را در مکانی قرار می دهیم که تمام عناصر سمت راست آن از عنصر لولا بزرگتر باشند و تمام عناصر سمت چپ عنصر لولا از آن کوچکتر باشند.
  - ۲-اگر مرتب سازی نزولی باشد عنصر لولا را در مکانی قرار می دهیم که تمام عناصر سمت راست آن از عنصر لولا کوچکتر باشند و تمام عناصر سمت چپ لولا از عنصر لولا بزرگتر باشند.
- پس از مشخص شدن مکان مناسب عنصر لولا آن گاه عناصر سمت چپ لولا و سمت راست لولا را به روش مرتب سازی سریع مرتب می کنیم.

الگوریتم های مرتب سازی

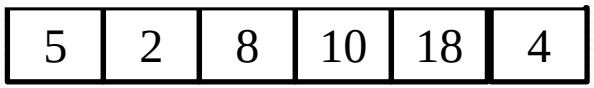
- مرتب سازی سریع quick sort

5	2	8	10	18	4

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort

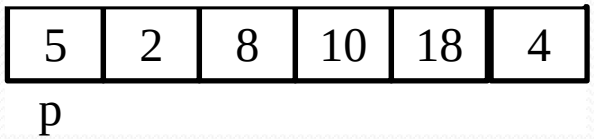
سمت چپ ترین عنصر را به عنوان
عنصر لولا انتخاب می کنیم



الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort

سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم



الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort

5	2	8	10	18	4

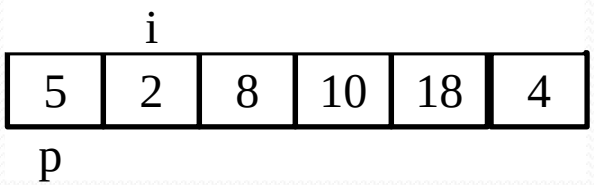
p

سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort

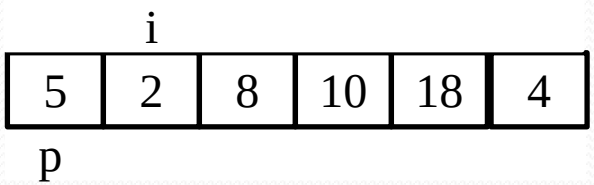


سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



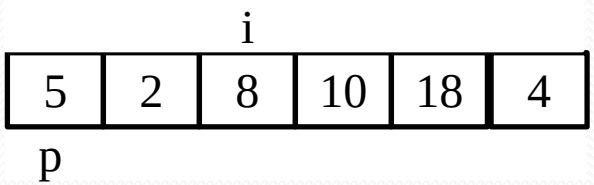
سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



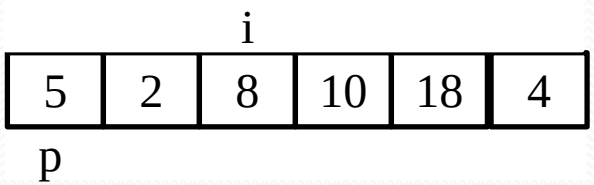
سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

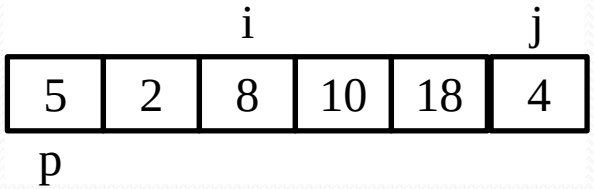
اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

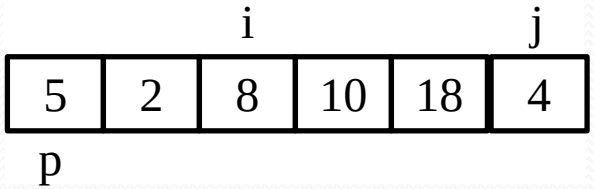
اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

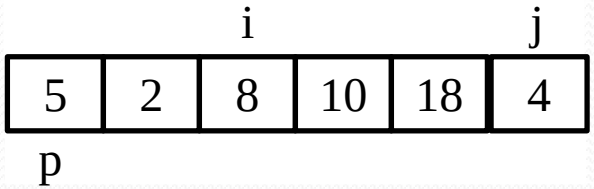
i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

j را آنقدر کاهش می دهیم تا a[j]<a[p] بشود

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

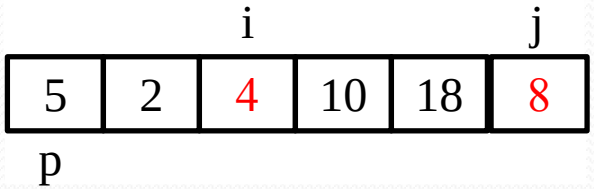
اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

j را آنقدر کاهش می دهیم تا a[j]<a[p] بشود

محتوای a[i] و a[j] را با هم تعویض می کنم(ااگر i>=j)

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

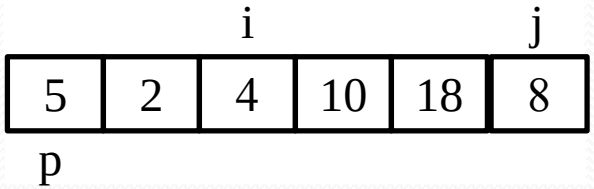
اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

j را آنقدر کاهش می دهیم تا a[j]<a[p] بشود

محتوای a[i] و a[j] را با هم تعویض می کنم(ااگر i>=j)

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

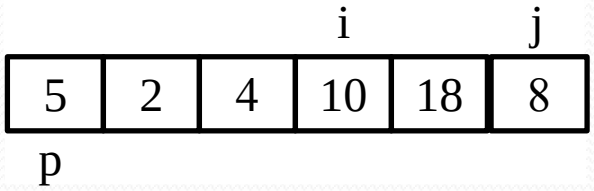
اندیس i را یکی بیشتر از اندیس p می گذارم

این عملیات را آنقدر تکرار می کنیم تا j<i بشود

- i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود
- اندیس j را برابر اندیس بالای آرایه قرار ی دهیم
- j را آنقدر کاهش می دهیم تا a[p]<a[j] بشود
- محتوای a[i] و a[j] را با هم تعویض می کنم(اگر i>=j)

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

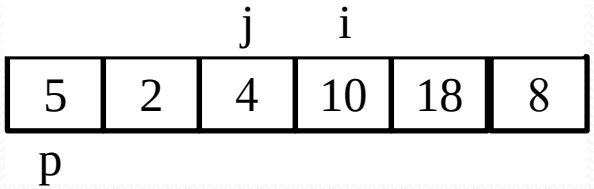
اندیس i را یکی بیشتر از اندیس p می گذارم

این عملیات را آنقدر تکرار می کنیم تا j<i بشود

- i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود
- اندیس j را برابر اندیس بالای آرایه قرار ی دهیم
- j را آنقدر کاهش می دهیم تا a[p]>a[j] بشود
- محتوای a[i] و a[j] را با هم تعویض می کنم(اگر i>=j)

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

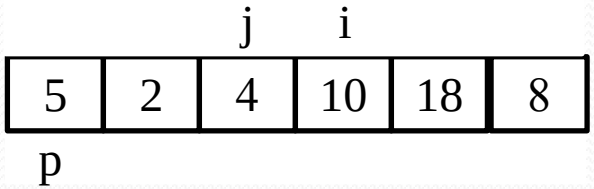
اندیس i را یکی بیشتر از اندیس p می گذارم

این عملیات را آنقدر تکرار می کنیم تا j<i بشود

- i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود
- اندیس j را برابر اندیس بالای آرایه قرار ی دهیم
- j را آنقدر کاهش می دهیم تا a[p]<a[j] بشود
- محتوای a[i] و a[j] را با هم تعویض می کنم(االگر i>=j)

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

j را آنقدر کاهش می دهیم تا a[j]<a[p] بشود

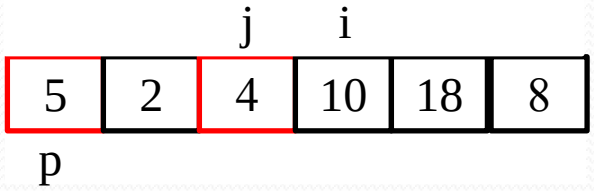
محتوای a[i] و a[j] را با هم تعویض می کنم(اگر i>=j)

این عملیات را آنقدر تکرار می کنیم تا j<i بشود

محتوای a[p] و a[j] را با هم تعویض می کنم

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

j را آنقدر کاهش می دهیم تا a[j]<a[p] بشود

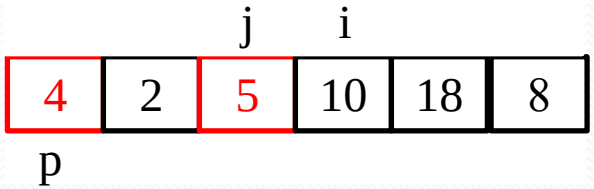
محتوای a[i] و a[j] را با هم تعویض می کنم(اگر i>=j)

این عملیات را آنقدر تکرار می کنیم تا j<i بشود

محتوای a[p] و a[j] را با هم تعویض می کنم

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

j را آنقدر کاهش می دهیم تا a[j]<a[p] بشود

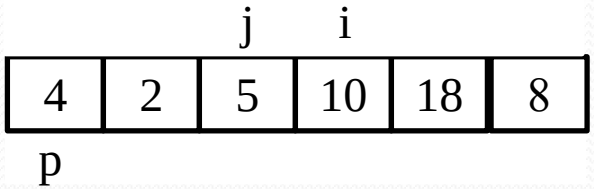
محتوای a[i] و a[j] را با هم تعویض می کنم(اگر i>=j)

این عملیات را آنقدر تکرار می کنیم تا j<i بشود

محتوای a[p] و a[j] را با هم تعویض می کنم

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



سمت چپ ترین عنصر را به عنوان عنصر لولا انتخاب می کنیم

اندیس i را یکی بیشتر از اندیس p می گذارم

i را آنقدر افزایش می دهیم تا a[i]>a[p] بشود

اندیس j را برابر اندیس بالای آرایه قرار ی دهیم

j را آنقدر کاهش می دهیم تا a[j]<a[p] بشود

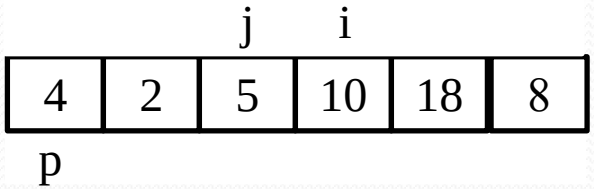
محتوای a[i] و a[j] را با هم تعویض می کنم(اگر i>=j)

این عملیات را آنقدر تکرار می کنیم تا j<i بشود

محتوای a[p] و a[j] را با هم تعویض می کنم

الگوریتم های مرتب سازی

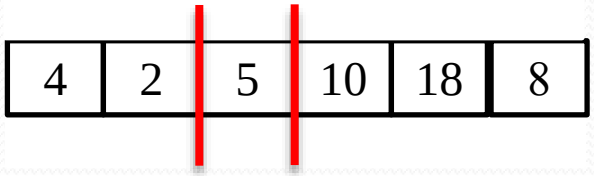
- مرتب سازی سریع quick sort



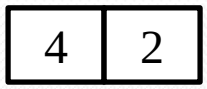
عملیات مرتب سازی سریع را برای عناصر قبل از j و عناصر بعد از j انجام ی دهیم

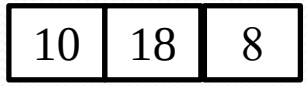
الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



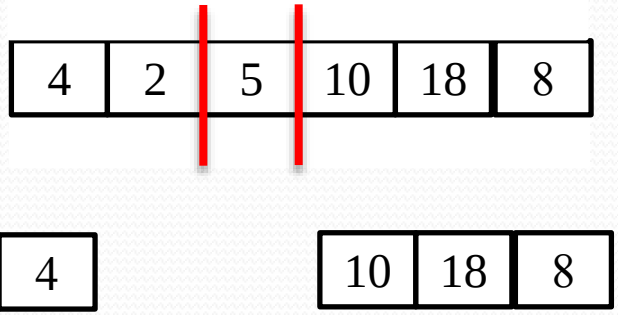
عملیات مرتب سازی سریع را برای عناصر قبل از j و عناصر بعد از j انجام ی دهیم





الگوریتم های مرتب سازی

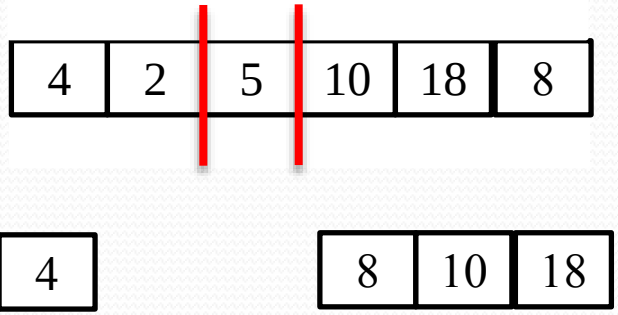
- مرتب سازی سریع quick sort



عملیات مرتب سازی سریع را برای عناصر قبل از j و عناصر بعد از j انجام ی دهیم

الگوریتم های مرتب سازی

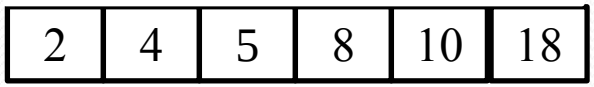
- مرتب سازی سریع quick sort



عملیات مرتب سازی سریع را برای عناصر قبل از j و عناصر بعد از j انجام ی دهیم

الگوریتم های مرتب سازی

- مرتب سازی سریع quick sort



√

الگوریتم های مرتب سازی

- مرتبه اجرایی quick sort

```
void quick_sort(int a[], int left, int right)
{
        if (left<right)
        {
                int i = left, j = right + 1, pivot = a[left];
                do
                {
                        do
                        { i++;      } while (a[i]<pivot);
                        do
                        {j--;       } while (a[j]>pivot);
                        if (i<j)
                                swap (a[i], a[j]);
                } while (i<j);
                swap (a[left], a[j]);
                quick_sort(a, left, j - 1);
                quick_sort(a, j + 1, right);
        }
}
```

O(n)

دو بار فراخوانی quick sort

الگوریتم های مرتب سازی

- مرتبه اجرایی quick sort
- در بهترین حالت و حال متوسط
- انتظار داریم که هر فراخوانی دوباره quick sort به اندازه تقریبا نصف آرایه اصلی باشد

$$T(n)=\begin{cases}1 & n==1\\ 2T(\frac{n}{2})+O(n) & else\end{cases}$$

a=2
b=2
$\theta(f(n))=O(n)$

$$\text{مرتبه اجریی}=\begin{cases}\theta\left(n^{log_2^2}\right) & \theta\left(n^{log_2^2}\right)>n\\ n & \theta\left(n^{log_2^2}\right)<n\\ \theta(n*\log n) & \theta\left(n^{log_2^2}\right)=n \quad \surd\end{cases}$$

الگوریتم های مرتب سازی

- مرتبه اجرایی quick sort
- بدترین حالت
- زمانی اتفاق می افتد که آرایه از قبل مرتب باشد. در این حالت یک سمت عنصر لولا عنصری وجود ندارد و در سمت دیگر آن تعداد n-1 عنصر وجود دارد

$$T(n)=\begin{cases}1 & n=1\\ T(n-1)+n & n>1\end{cases}$$

$$T(n)=T(n-1)+n=T(n-2)+(n-1)+n=\dots=1+2+3+\dots+(n-1)+n=\frac{n(n+1)}{2}\rightarrow O(n^2)$$