Data structure

DR RASTGOO

فصل ۱۰:

مرتب سازی

مرتب سازی

الگوریتم های مرتب سازی را از نظر نحوه مرتب سازی داده ها می توان به صورت زیر دسته بندی کرد:

۱- مرتب سازی مقایسه ای

در مرتب سازی مقایسه ای، اطلاعات دیگری از داده های ورودی وجود ندارد و فقط با مقایسه بین کلیدهای عناصر، ترتیب نسبی آنها پیدا می شود.

۲- مرتب سازی غیر مقایسه ای(خطی)

در مرتب سازی غیر مقایسه ای، بدون مقایسه کلیدهای عناصر با هم، عمل مرتب سازی انجام می شود.

نام روش	بهترین	میانگین	بدترین
Bubble	$o(n)$	$o(n^2)$	$o(n^2)$
Selection	$o(n)$	$o(n^2)$	$o(n^2)$
Insertion	$o(n)$	$o(n^2)$	$o(n^2)$
Quick	$o(n.\log n)$	$o(n.\log n)$	$o(n^2)$
Merge	$o(n.\log n)$	$o(n.\log n)$	$o(n.\log n)$
Heap	$o(n.\log n)$	$o(n.\log n)$	$o(n.\log n)$
Tree	$o(n.\log n)$	$o(n.\log n)$	$o(n^2)$

مرتب سازی حبابی (Bubble Sort)

در این روش با n بار حرکت در طول آرایه، یک عنصر با عنصر بعدی مقایسه شده و در صورت لزوم جا به جا می‌شوند.

در مرتبه اول طی کردن آرایه، بزرگ‌ترین (یا کوچکترین) عنصر در انتهای آرایه قرار می‌گیرد .

این مرتب سازی ، در i امین مرتبه، عناصر n – i تا n به طور صحیح قرار می‌گیرند.

مثال

اعداد 2 , 3 , 8 , 4 را به روش حبابی، صعودی نمایید.

مقایسه ۴ با ۸ : 2 , 3 , 8 , 4

مقایسه ۳ با ۸ و تعویض آنها : 2 , 8 , 3 , 4 گذر اول

مقایسه ۸ با ۲ و تعویض آنها : 8 , 2 , 3 , 4

در پایان گذر اول ، بزرگترین عنصر در خانه آخر قرار گرفته است.

گذر دوم

مقایسه ۴ با ۳ و تعویض آنها : 3 , 4 , 2 , 8

مقایسه ۴ با ۲ و تعویض آنها : 3 , 2 , 4 , 8

گذر سوم

مقایسه ۳ با ۲ و تعویض آنها : 2 , 3 , 4 , 8

عناصر داده شده بعد از ۳ گذر با ۶ مقایسه که ۵ تا از آنها منجر به تعویض شد، مرتب شدند.

- تعداد مقایسه ها در روش مرتب سازی حبابی برابر $\frac{n(n-1)}{2}$ می باشد که در بدترین حالت به همین تعداد، تعویض انجام می گیرد.

- بزرگترین عنصر در مرتب سازی حبابی صعودی بعد از n-1 مقایسه و حداکثر n-1 تعویض به انتهای لیست می رود.

الگوریتم مرتب سازی حبابی

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

مرتب‌سازی انتخابی (Selection Sort)

در مرتب‌سازی صعودی انتخابی، ابتدا کوچک‌ترین عنصر آرایه را پیدا کرده و آن را در مکـان اول لیسـت قـرار می‌دهـد.

آنگـاه

کوچکترین عنصر دوم داخل لیست را پیدا کرده و آن را در مکان دوم لیست قرار می‌دهد و ...

مثال

تعداد مقایسه های لازم برای مرتب کردن لیست زیر به روش انتخابی را محاسبه نمایید.

10 , 30 , 17 , 12 , 1 , 21 , 15

6+5+4+3+2+1=21

لیست اولیه	**10**	**30**	**17**	**12**	**1**	**21**	**15**
مرحله اول	**1**	**30**	**17**	**12**	**10**	**21**	**15**
مرحله دوم	**1**	**10**	**17**	**12**	**30**	**21**	**15**
مرحله سوم	**1**	**10**	**12**	**17**	**30**	**21**	**15**
مرحله چهارم	**1**	**10**	**12**	**15**	**30**	**21**	**17**
مرحله پنجم	**1**	**10**	**12**	**15**	**17**	**21**	**30**
مرحله ششم	**1**	**10**	**12**	**15**	**17**	**21**	**30**

مثال

تعداد مقایسه های لازم برای مرتب کردن لیست زیر به روش انتخابی را محاسبه نمایید.

10 , 30 , 17 , 12 , 1 , 21 , 15

6+5+4+3+2+1=21

لیست اولیه	10	30	17	12	1	21	15
مرحله اول	1	30	17	12	10	21	15
مرحله دوم	1	10	17	12	30	21	15
مرحله سوم	1	10	12	17	30	21	15
مرحله چهارم	1	10	12	15	30	21	17
مرحله پنجم	1	10	12	15	17	21	30
مرحله ششم	1	10	12	15	17	21	30

مثال

تعداد مقایسه های لازم برای مرتب کردن لیست زیر به روش انتخابی را محاسبه نمایید.

10 , 30 , 17 , 12 , 1 , 21 , 15

6+5+4+3+2+1=21

لیست اولیه	**10**	**30**	**17**	**12**	**1**	**21**	**15**
مرحله اول	**1**	**30**	**17**	**12**	**10**	**21**	**15**
مرحله دوم	**1**	**10**	**17**	**12**	**30**	**21**	**15**
مرحله سوم	**1**	**10**	**12**	**17**	**30**	**21**	**15**
مرحله چهارم	**1**	**10**	**12**	**15**	**30**	**21**	**17**
مرحله پنجم	**1**	**10**	**12**	**15**	**17**	**21**	**30**
مرحله ششم	**1**	**10**	**12**	**15**	**17**	**21**	**30**

مثال

تعداد مقایسه های لازم برای مرتب کردن لیست زیر به روش انتخابی را محاسبه نمایید.

10 , 30 , 17 , 12 , 1 , 21 , 15

6+5+4+3+2+1=21

لیست اولیه	**10**	**30**	**17**	**12**	**1**	**21**	**15**
مرحله اول	**1**	**30**	**17**	**12**	**10**	**21**	**15**
مرحله دوم	**1**	**10**	**17**	**12**	**30**	**21**	**15**
مرحله سوم	**1**	**10**	**12**	**17**	**30**	**21**	**15**
مرحله چهارم	**1**	**10**	**12**	**15**	**30**	**21**	**17**
مرحله پنجم	**1**	**10**	**12**	**15**	**17**	**21**	**30**
مرحله ششم	**1**	**10**	**12**	**15**	**17**	**21**	**30**

مثال

تعداد مقایسه های لازم برای مرتب کردن لیست زیر به روش انتخابی را محاسبه نمایید.

10 , 30 , 17 , 12 , 1 , 21 , 15

6+5+4+3+2+1=21

لیست اولیه	**10**	**30**	**17**	**12**	**1**	**21**	**15**
مرحله اول	**1**	**30**	**17**	**12**	**10**	**21**	**15**
مرحله دوم	**1**	**10**	**17**	**12**	**30**	**21**	**15**
مرحله سوم	**1**	**10**	**12**	**17**	**30**	**21**	**15**
مرحله چهارم	**1**	**10**	**12**	**15**	**30**	**21**	**17**
مرحله پنجم	**1**	**10**	**12**	**15**	**17**	**21**	**30**
مرحله ششم	**1**	**10**	**12**	**15**	**17**	**21**	**30**

مثال

تعداد مقایسه های لازم برای مرتب کردن لیست زیر به روش انتخابی را محاسبه نمایید.

10 , 30 , 17 , 12 , 1 , 21 , 15

6+5+4+3+2+1=21

لیست اولیه	**10**	**30**	**17**	**12**	**1**	**21**	**15**
مرحله اول	**1**	**30**	**17**	**12**	**10**	**21**	**15**
مرحله دوم	**1**	**10**	**17**	**12**	**30**	**21**	**15**
مرحله سوم	**1**	**10**	**12**	**17**	**30**	**21**	**15**
مرحله چهارم	**1**	**10**	**12**	**15**	**30**	**21**	**17**
مرحله پنجم	**1**	**10**	**12**	**15**	**17**	**21**	**30**
مرحله ششم	**1**	**10**	**12**	**15**	**17**	**21**	**30**

- در مرتب سازی انتخابی، اعداد بعد از n-1 مرحله مرتب می شوند.

- تعداد مقایسه ها در مرتب سازی انتخابی برابر $\frac{n(n-1)}{2}$ می باشد.

- تعداد جابجایی ها در مرتب سازی انتخابی در بدترین حالت برابر $\frac{n(n-1)}{2}$ می باشد. (بـدترین حالـت وقتی رخ می دهد که بخواهیم آرایه صعودی را نزولی کنیم و یا بر عکس)

الگوریتم مرتب سازی انتخابی

```
void   selectionsort (int  s[ ] , int n){
    int   i , j , min;
    for ( i = 0; i< n-1; i++){
          min = i;
          for (j = i+1; j <= n ; j++)
              if (s[j] < s[min])
                    min = j;
          swap(s[i] , s[min]);
     }
}
```

مرتب‌سازی درجی (Insertion Sort)

این الگوریتم بر اساس درج یک عنصر در محل صحیح کار می‌کند.

برای آرایه A با n عنصر:

۱ـ A[1] مرتب است.

۲ـ A[2] قبل یا بعد از A[1] درج می‌شود. طوری که A[1] و A[2] مرتب باشند.

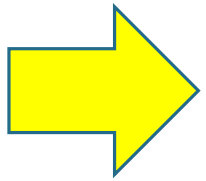
۳ـ A[3] در مکان صحیح خود نسبت به A[1] و A[2] درج می‌شود، طوری که A[1],A[2],A[3] مرتب باشند.

۴ـ الی آخر

مثال

آرایه زیر را به روش درجی مرتب سازید.

20 , 5 , 35 , 8 , 2



20					درج ۲۰
5	**20**				درج ۵
5	**20**	**35**			درج ۳۵
5	**8**	**20**	**35**		درج ۸
2	**5**	**8**	**20**	**35**	درج ۲

الگوریتم مرتب سازی درجی

```
void    insertionsort (int  s[ ] , int   n){
      int   i,j; int  x;
      for (i=1 ; i<n ; i++){
            x = s[i];
            for( j = i-1 ;  j>=0  &&  x < s[j] ;  j--)
                  s[j+1]=s[j];
            s[j+1]=x;
      }
```

- مرتب سازی درجی، برای یک آرایه مرتب، بهترین عملکرد و برای یک آرایه مرتب معکوس، بدترین عملکرد را دارد.

- مرتب سازی درجی برای n<=20 ، سریع‌ترین روش مرتب‌سازی است.

- اگر n بزرگ باشد و رکوردها هم بزرگ باشند (زمان انتساب آنها چشمگیر باشد)، مرتب سازی انتخابی باید بهتر از مرتب سازی درجی عمل کند.

در مرتب سازی درجی، یک آرایه **n** عنصری را می توان با **n-1** عمل درج مرتب کرد.

اگر از جستجویی دودویی استفاده کنیم، تعداد مقایسه‌ها کاهش می‌یابد. این روش را مرتب‌سازی **درجی دودویی** می‌نامند.

وارونگی

وارونگـی در یـک جایگشـت، زوج (k_i, k_j) اسـت بـه طـوری کـه $i < j$ و $k_i > k_j$ باشـد.

بـه طـور مثـال، در جایگشـت [3,2,4,1,6,5] ، زوج (3,2) یک وارونگی است، چون 2>1 و 3>2 .

وارونگی های این جایگشـت عبارتنـد از: , (3,1) , (3,2)(6,5) , (4,1) , (2,1).

یک جایگشت، وارونگی نخواهد داشت اگر و فقط اگر دارای ترتیب مرتب [1,2,…,n] باشد.

مرتب سازی ادغام (Merge Sort)

در این روش آرایه n عنصری به n قسمت به طول یک تقسیم و سپس هر دو قسمت مجاور به صورت مرتب شده با هم ادغام می‌شوند.

در این حالت آرایه به طول ۲ ایجاد شده است.

روند ادغام تا رسیدن به یک آرایه به طول n ادامه می‌یابد.

مثال

دو آرایه مرتب A و B را ادغام کرده و حاصل را در آرایه C قرار دهید:

A : 1, 5 , B : 2, 7, 9, 20

حل: برای اینکار از ۳ متغیر اشاره گر pa وpb وpc استفاده می شود که ۲ تای خاص از آنها با یکدیگر تغییر می کنند. در ابتدا pa به مقدار ۱ در آرایه A و pb به مقدار ۲ در آرایه B وpc به ابتدای آرایه خالی C اشاره می کند. محتـوای pa و pb بـا هـم مقایسه شده و چون مقدار ۱ کوچکتر است در آرایه C نوشته شده و اشاره گر pa و pc یک واحد اضافه شـده و مقـدار ۵ بـا ۲ مقایسه شده و چون ۲ کوچکتر است در آرایه C نوشته شده و pb و pc یک واحد اضافه می شوند .حال مقدار ۵ بـا ۷ مقایسـه شده و مقدار ۵ در آرایه c نوشته شده و در این حالت چون به انتهای آرایه A رسیده ایم بقیه عناصر آرایه B را به انتهای آرایه C اضافه می کنیم: C : 1 , 2 , 5 , 7 , 9 , 20

مثال

لیست داده شده در روش Merge Sort ، بعد از چند گذر مرتب می شود؟

2 , 3 , 1, 7 , 6 , 5, 4, 0

[2] [3] [1] [7] [6] [5] [4] [0]
[2,3] [1,7] [5,6] [0,4]
[1,2,3,7] [0,4,5,6]
[0,1,2,3,4,5,6,7]

در مرتب سازی ادغام ، مراحل کار به صورت زیر می باشد:

۱- تقسیم لیست n عنصری به دو زیر لیست $\frac{n}{2}$ عنصری.

۲- مرتب کردن بازگشتی دو زیر لیست به روش مرتب سازی ادغام در زمان $2T(\frac{n}{2})$.

۳- ادغام دو زیر لیست مرتب شده در زمان $\theta(n)$.

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

$$o(n.\log n)$$

مرتب‌سازی سریع (Quick Sort)

یک عنصر به عنوان محور انتخاب شده(معمولا عنصر اول) و سایر عناصر در آرایه به صورتی جا به جا می‌شوند که کلیه عناصر کوچکتر از محور در یک طرف و کلیه عناصر بزرگ‌تر از محور در طرف دیگر قرار گیرند، سپس به همین روش دو آرایه واقع در طرفین محور به صورت بازگشتی مرتب می‌شوند.

(a) : هیچ یک از عناصر تقسیم بندی نشده اند.

(b) : مقدار 2 با خودش تعویض و در قسمت مقادیر کوچکتر قرار گرفته است.

(c) : مقدار 8 به قسمت مقادیر بزرگتر اضافه شده است.

(d) : مقدار 7 به قسمت مقادیر بزرگتر اضافه شده است.

(e) : مقادیر 1 و8 تعویض شده اند و قسمت کوچکتر افزایش می یابد.

(f) : مقادیر 3 و8 عوض شده و قسمت کوچکتر افزایش می یابد.

(g) : مقدار 5 به قسمت بزرگتر اضافه شده است.

(h) : مقدار 6 به قسمت بزرگتر اضافه شده است و حلقه پایان می یابد.

(i) : عنصر محوری عوض شده تا بین دو قسمت قرار بگیرد.

مرتب سازی درختی (Tree Sort)

ابتدا با اعداد داده شده یک BST ساخته و سپس با پیمایش میانوندی درخت حاصل، رشته صعودی حاصل می شود.