بسم الله الرحمن الرحيم

موضوع پروژه :

Merge Sort

نام تهیه کننده :
حمید امید

نام استاد مربوطه :
جناب آقاي سهامي

با تشکر وسپاس فراوان

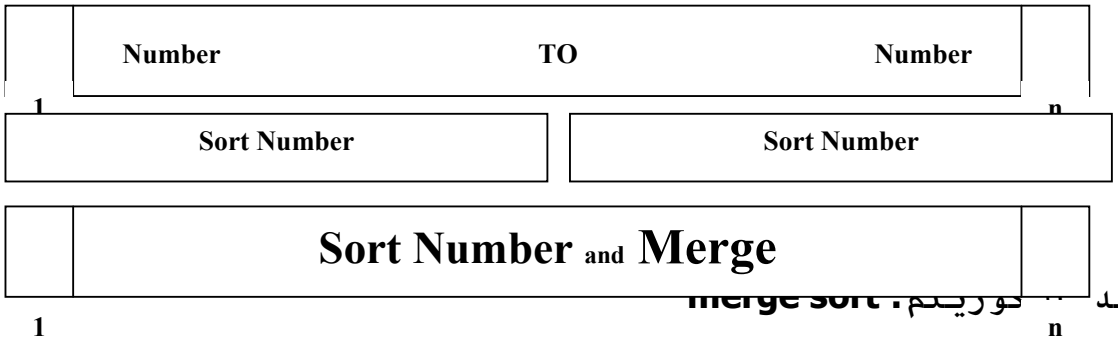
مرتب سازی به روش **merge** یا ادغام

اصول کار:
اگر مجموعه ای از اعداد داشته باشیم و آن را به **2** بخش تقسیم کنیم و هر بخش را جداگانه مرتب کنیم و حاصل را به گونه ای ادغام کنیم که این ترتیب در آن مجموعه حاصله هنوز مرتب باشد در این حالت کل مجموعه مرتب شده به حساب می آید.

یه سوال که مطرح میشه اینه که خوب حالا به فرض اونو به **2** بخش تقسیم کردیم - هر بخش به چه صورت مرتب شوند؟
جواب : مشخص است هر بخش نیز به همان صورت مرتب میشود (**2** تیکه شدن). یعنی هر بخش کوچکتر هم باید کوچکتر شود و مرتب الی آخر.
یه سوال دیگر : این تیکه تیکه کردن داده های خانه آرایه مثلا تا کجا ادامه پیدا کنه ؟
خوب مسلمه که اگر به عنصر تک عضوی رسیدیم دیگه نمیشه اونو **2** تیکه کرد. پس کار خرد شدن ادامه پیدا میکنه تا به یه عنصر (مثلا یک عدد در خانه ای ازخانه های حافظه) برسیم.

ببینید سعی میکنیم این **2** تیکه کردن ها مساوی باشند) در هر طرف تقسیم تعداد مساوی خانه حافظه یا آرایه قرار گرفته شده باشد) - دلیلی نداره مساوی تقسیمشون نکنه وقتی....

مراحل کار رو به صورت زیر ادامه میدهیم:
-اگر تعداد داده ها یک یا کمتر شد دیگر نیازی به مرتب سازی نیست و برگرد.
-وسط داده ها را پیدا کن.
-نیمه اول داده ها را به روش **Merge Sort** مرتب کن.
-نیمه دوم داده ها را به روش **Merge Sort** مرتب کن.
-دو نیمه مرتب شده را به گونه ای ادغام کن که حاصل مرتب باشد!



کد ... الگوریتم **merge sort** :

کد:

```
void mergesort (int low,int high)
{
int mid;
if(low<high){
mid=(low+high)/2;
mergesort(low,mid)
mergesort(mid+1,high);
merge(low,mid,high);
}
```

زیر الگوریتم **merge** که در خط آخر الگوریتم بالا میبینید استفاده شده کارش اینه که دو بخش مرتب شده آرایه را با هم ادغام کند به گونه ای که حاصل هم مرتب باشد.

فرض کنید شما به تعداد **N** تا عدد از کاربر با یه حلقه **for** دریافت کردین و اونو تو یه آرایه گذاشتین که اسم آرایه هم **A** هست.
کد:

```
mergesort(low,mid)
```

این کد بالا شامل بخش اول آرایه **A** از محل **Low** تا **Mid** و بخش دوم آن از **Mid+1** تا **high** را شامل میشود.
نکته : چون همل ادغام را نمیشود به صورت درجا انجام داد (نمیشود همون آرایه که داریم روی آن کار میکنیم بدون خانه حافظه کمکی هم از آن بخ.انیم هم آن را ادغام کنیم(بنابراین نیاز به یک آرایه کمکی هست به نام **B** از نوع **A** و با همان اندازه.**A**

یه مثال میزنم:
آرایه ای **10** عنصری به نام **A** دارم که عناصر **0** تا **4** آن مرتب و عناصر **5** تا **9** هم مرتب هستند. میخواهیم این دوبخش را ادغام کرده و در آرایه **B** قرار دهیم.

کد:

```
A={350,365,390,500,700,340,400,450,490,495}
```

در این وضعیت **low=0 , mid=4 , mid+1=5 , high=9** میباشند.

برای ادامه کار سه متغییر به نامهای **h,i,j** تعریف میکنم و نقش های زیر را به آنها میدهیم:
کد:

```
h=low
```

محلی از اولین بخش آرایه **A** که داده آن باید بررسی گردد.

کد:

```
j=mid+1
```

محلی از دومین بخش آرایه **A** که داده آن باید بررسی گردد.

کد:

```
i=low
```

محلی از آرایه **B** که داده جدید باید در آن ریخته شود.
کار : کافی است که داده محل **h** و داده محل **j** آرایه **A** را با هم مقایسه کنیم و آن را که کوچکتر است در محل **i** آرایه **B** قرار دهیم.
سپس اگر داده محل **h** کوچکتر بوده است به **h** یکی اضافه شود و اگر داده محل **j** کوچکتر بوده است به **j** یکی اضافه شود.
اگر **2** داده مساوی بودند فرقی ندارد که کدامیک انتخاب شود.
وچون در آرایه **B** هم عددی درج شده باید به **i** یک واحد اضافه شود.عملیات را با **h,i,j** جدید ادامه میدهیم تا یکی از **2** بخش آرایه **A** تمام شود. مشخص است که بقیه داده های بخش نا تمام آرایه **A** باید به همان ترتیب به آرایه **B** منتقل شوند.

ادامه مثال بالا:

LEFT]

کد:

```
A={350,365,390,500,700,340,400,450,490,495}
```

[/left]

h=0 --- j=5

LEFT]

کد:

```
B={   ,   ,   ,   ,   ,   ,   ,   ,   ,   }
```

[/left]

i=0

پس از انجام اولین مقایسه نتیجه این گونه خواهد بود:

LEFT]

کد:

```
A={350,365,390,500,700,340,400,450,490,495}
```

[/left]

h=0 --- j=6

LEFT]

کد:

```
B={340,   ,   ,   ,   ,   ,   ,   ,   ,   }
```

[/left]
i=1

پس از انجام دومین مقایسه نتیجه این گونه خواهد بود:
LEFT]
کد:

```
A={350,365,390,500,700,340,400,450,490,495}
```

[/left]
h=1 --- j=6
LEFT]
کد:

```
B={340,350,   ,   ,   ,   ,   ,   ,   ,   }
```

[/left]
i=2

اگر این عملیات را برای 6 بار انجام دهید نتیجه زیر بدست می آید:
LEFT]
کد:

```
A={350,365,390,500,700,340,400,450,490,495}
```

[/left]
h=3 --- j=10
LEFT]
کد:

```
B={340,350,365,390,400,450,490,496,   ,   }
```

[/left]
i=8

در این مرحله بخش دوم آرایه تمام شده است و اعداد باقیمانده بخش اول که 500 و 700 میباشد باید به B منتقل شوند و حاصل به صورت زیر خواهد بود:
LEFT]
کد:

```
B={340,350,365,390,400,450,490,496,500,700}
```

[/left]

حال کافی است که داده های آرایه B را به همان ترتیب به مکان A منتقل کنیم!

حالا الگوریتم **merge** رو مینویسیم:

کد:

```
void mege(int low, int mid, int high)
{
int h,i,j,k;
int B[];
h=low;
i=low;
j=mid+1;
while(h<=mid && j<=high){
if(A[h]<=A[j]){
B[i]=A[h];
h++;
}
else{
B[i]=A[j];
j++;
}
i++;
}
if(h>mid)
for (k=j;k<=high;k++){
B[i]=A[k];
i++;
}
else
for(k=h;k<=mid;k++){
B[i]=A[k];
i++;
}
for(k=low;k<=high,k++)
A[k]=B[k];
}
```

برای استفاده از **mergesort** پس از دریافت داده ها و ذخیره سازی آنها در **A** الگوریتم را به صورت زیر اجرا کنید:

کد:

```
mergesort(0,n-1)
```

پیچیدگي زماني **Merge sort** :

که توضیحات آن بصورت مفصل در قسمتهاي بالا گفته شد میشود :
پیچیدگي اوغامي +T(n) +2t(n)

W(n)=n-1

Log2=1
F(n)=n-1=ɵ(n)
➔ W(n)= ɵ(n*lg n)
= ɵ(n*lg n)

که توضیحات آن در مباحثه بالا ذکر شد .