

# 算法导论

---

## Chapter 1 分而治之篇

---

### 1.1 归并排序

(又称分而治之)

#### 1.1.1 选择排序

- 每次从序列中选择最小(大)的元素排序

#### 1.1.2 插入排序

- 依次将每个元素插入到已排序的序列之中

#### 1.1.3 问题背景

- 序列特点:局部有序,
- 时间复杂度: 与分组p有关, 分组p代表着原本序列的顺序度(详见如下图)

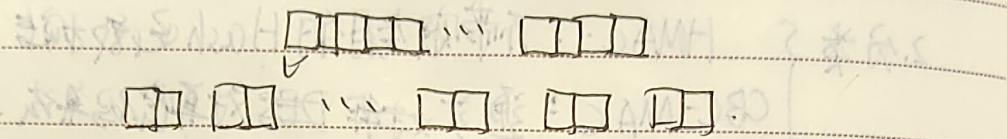


华中科技大学

HUZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

www.hust.edu.cn

归并排序



设数组长为  $n$ , 有  $n = P \times q$   
且  $P \geq 1$  &  $q \geq 1$  有序的每组长度  $\downarrow$   $q$  组  $\rightarrow$  P 改变  
 $\rightarrow$  常数

假设两两一组分组排序, 有如下迭代: (假设最小用时)

有	第 <i>i</i> 次	每组长度	组数	比较次数
	1	$P$	$q$	$P \times \frac{q}{2}$
	2	$\frac{q}{2}$	$2P$	$2P \cdot \frac{q}{2 \cdot 2}$

$$\begin{aligned} \text{有 } q \cdot 2^{1-i} &= 2 \\ 2^{1-i} &= \frac{2}{q} \\ 2^1 &= q \\ i &= \log_2 q \end{aligned}$$

$$\begin{aligned} \frac{Pq}{2} &\xrightarrow{\text{常数}} \frac{Pq}{2} \\ \frac{Pq}{2} &\xrightarrow{\text{常数}} \frac{Pq}{2} \cdot \frac{2}{2} \\ \frac{Pq}{2} &\xrightarrow{\text{常数}} \frac{Pq}{2} \cdot \frac{2}{2} \end{aligned}$$

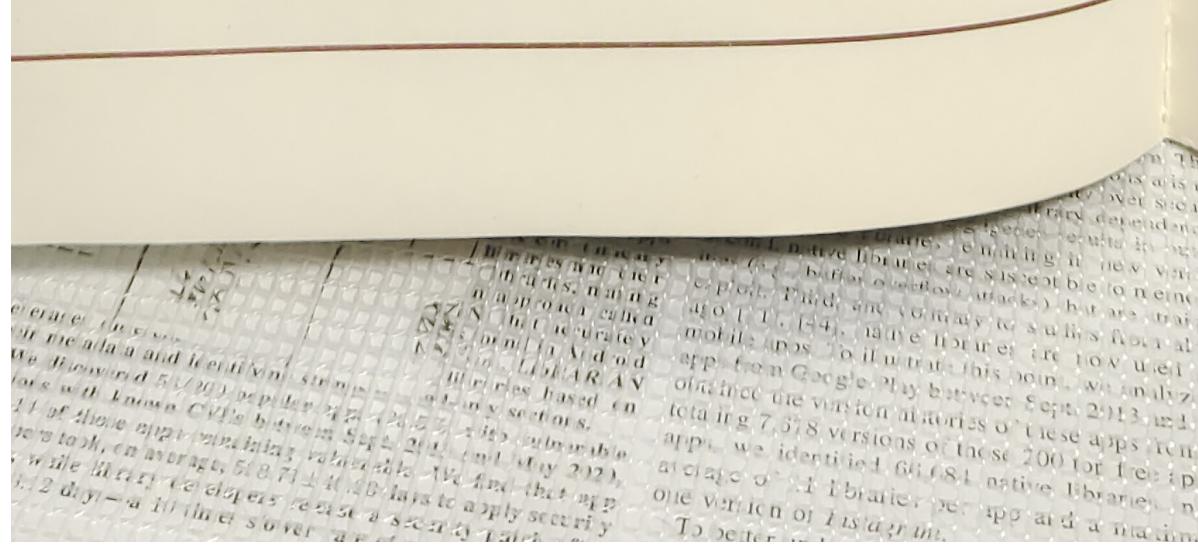
时间复杂度为  $O(n) = \frac{Pq}{2} \cdot i = \frac{n}{2} \cdot (\log_2 q)$

又: 与常数无关,  $\log_2$  可换为任意底常数

$\therefore O(n) = n \log q$ ,  $q \rightarrow 1 \Leftrightarrow$  整个数组有序  $\Leftrightarrow O(n) \rightarrow 0$ ,  
 $q \rightarrow n \Leftrightarrow$  无序  $\Leftrightarrow O(n) = n \log^n$

归并排序复杂度:  $n(\log n)$

(也可用画二叉树求解)



伪代码如下

```
1 //input:A[n]    需要排序的数组
2 //output:A[n]   已排好的数组
3 //递归终止条件: left>right
4 void MergeSort(A,left,right){
5     if(left>=right)
6         return A[left..right];           //这时表明分组下只有一个元素了
7     MergeSort(A,left,(right+1)/2);    //先把左边分组排好
8     MergeSort(A,(right+1)/2+1,right); //再把右边分组拍好
9     Merge(A,left,right);            //左右分组合并排序
10    return A[left...right];
11    //return A;
12 }
13
14 int tmp[100000];
15 void Merge(A,left,right){
16     int mid=(left+right)/2;
17     int i=left,j=mid+1;
18     int *ptmp=tmp;
19     while( i<=mid && j<=right) //循环的停止条件, 标志着一个数组已经被遍历排序完
20     {
21         if(A[i]<=A[j]){
22             *ptmp++=A[i++];
23         }
24         else{
25             *ptmp++=A[j++];
26         }
27     }
28     if(i>mid){
29         while(j<=right){
30             *ptmp++=A[j++];
31         }
32     }
33     else{
34         while(i<=mid){
35             *ptmp++=A[i++];
36         }
37     }
38     memcpy(A[left],tmp,right-left+1);
```

具体代码请见 code\Mergecode.cpp

## 1.2 递归式求解

### 1.3 最大/最小子数组

#### 问题背景

- 输入：给定一个数组A[n],求最大和或最小和
- 输出: S(l,r), 即从l 到r 的数组元素之和S<sub>max</sub> (三个输出)

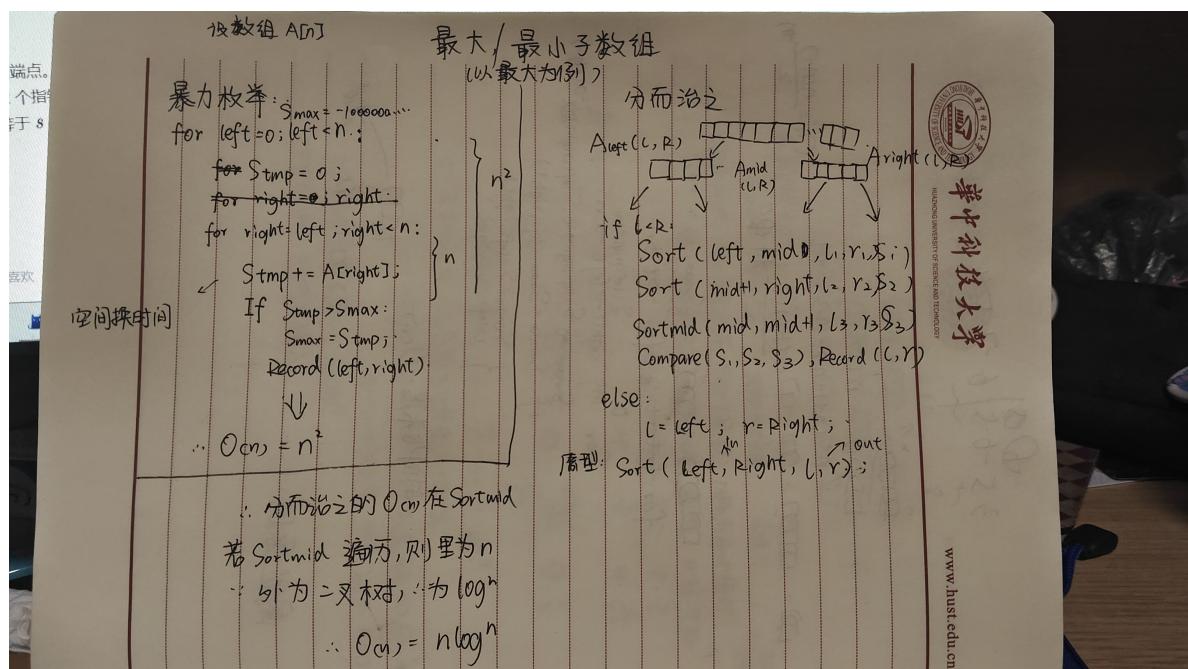
暴力枚举法：时间复杂度O( n<sup>2</sup> ), 伪代码如下:

```

1 int Smax=-10000000000000000; int Stmp;
2 for left =0->n:
3     Stmp=0;
4     for right=left->n:
5         Stmp+=A[right];
6         Compare(Stmp,Smax);
7

```

我们采用分而治之的策略，不考虑分的问题，考虑合的问题（思路如下）



所以写出如下伪代码:

```

1 //leftrecord用来记录当前求出的left
2 void Sort(int *a,int left,int right,int &leftrecord,int &rightrecord){
3     int mid=(left+right)/2;
4     int l1,l2,l3,r1,r2,r3,s1,s2,s3;
5
6     if(left>=right){
7         ;
8     }
9     else{

```

```

10     Sort(a, left, mid, l1, r1);
11     Sort(a, mid+1, right, l2, r2);
12     SubSort(a, left, right, l3, r3);
13     s1=s(l1, r1);
14     s2=s(l2, r2);
15     s3=s(l3, r3);
16     Compare(s1, s2, s3);
17 }
18 }
```

所以算法的主要时耗来源于SubSort(),即比较合并数组的最大/小子数组;

因为是合并, 所以必定从中间开始, 包含mid,mid+1项, 向两边扩张

```

1 void SubSort(int *a, int left, int &right, int l3, int r3){
2     int l, r, mid=(left+right)/2;
3     int Smax=-10000000000, Stmp=0;
4     for l=mid->0:
5         Stmp+=a[l];
6         Compare(Stmp, Smax);
7         Smax=-10000000000, Stmp=0;
8     for r=mid+1->right:
9         Stmp+=a[r];
10        Compare(Stmp, Smax);
11 }
```

外层为二叉树, 所以复杂度为 $\log n$ ,里层为遍历, 所以为 $n$ ,  $\therefore O(n)=n\log n$

代码见: minimum\_array.cpp

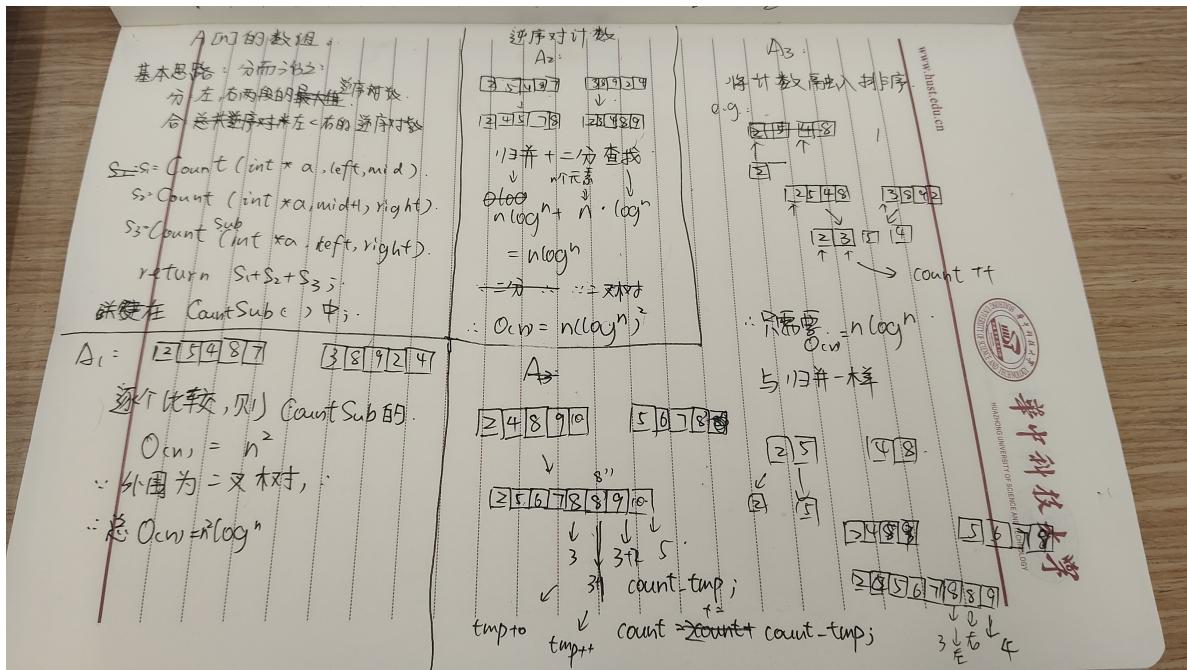
## 1.4 逆序对计数问题

总体思路还是分而治之, 大概伪代码如下:

```

1 int Count(int *a, int left, int mid){
2     if(left>=right)
3         return 0;
4     else
5         return (Count(a, left, mid)+Count(a, mid+1, right)+CountSub(a, left, right));
6 }
```

对于CountSub(), 有如下三种方案计算, 如下图所示:



对应的伪代码如下：

方法一。

```

1 int CountSub(int *a, int *left, int *right){
2     int mid=(left+right)/2, count=0;
3     for(int i=left, i<=mid; i++)
4         for(int j=mid+1; j<=right; j++)
5             if(a[i]>a[j]) count++;
6     }
7 }
```

方法二：

```

1 int CountSub(int *a, int *left, int *right){
2     int mid=(left+right)/2, count=0;
3     MergeSort(a, left, right);
4     for(int j=mid+1; j<=right; j++)
5         return quicksort();
6     }
7 }
```

方法三：

```

1 int tmp[10000];
2 int CountSub(int *a, int left, int right){
3     int i=left, j=mid+1;
4     *ptmp=tmp; int count=0, count_tmp=0;
5     while(i<=mid && j<=right){
6         if(A[i]<=A[j]){
7             *ptmp+++=A[i++];
8             count+=count_tmp;
9         }
10        else{
11            *ptmp+++=A[j++];
12            count_tmp++;
13        }
14    }
15    return count;
16 }
```

```
13     }
14 }
15 if(i>mid){
16     while(j<=right){
17         *ptmp++=A[j++];
18     }
19 }
20 else{
21     while(i<=mid){
22         *ptmp++=A[i++];
23         count+=count_tmp;
24     }
25     memcpy(tmp,a);
26 }
```

具体代码见reverse\_pair.cpp