

基础算法



南京航空航天大学
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS



MS · OI
李 斯 棋

Author : Makik
Presentation : Makik

目录

- 一、时空分析
- 二、枚举
- 三、序列算法选讲
- 四、二分、三分
- 五、递归
- 六、排序算法

斯.....
斯国一！

“



1

时空分析

我的程序跑地
比香港记者都快

“





时间复杂度



概念

在ACM范围里，我们只需了解时间复杂度可以大致地通过一个算法运算的次数来描述程序运行的效率，**常常用大写字母 O 来表示**。在表示时间复杂度的时候，**只保留数量级最大的一项，并忽略系数**。

举个栗子

对于给定的常数 N ，若某一个算法计算的次数是 $3N^3 + N^2 + 10^5$ ，则我们可以认为它的时间复杂度为 $O(N^3)$ 。





空间复杂度



概念

顾名思义就是用来衡量内存的占有量的。除了计算复杂度，我们常常也需要直接算出来运行程序需要占用多少内存。

超高校级的单位换算

1MB = 1024KB 1KB = 1024B 1B (byte, 字节)=8b (bit, 比特).

常见数据类型

char/bool/short. 1B *int/float*. 4B *long long/double*. 8B





时空复杂度



数量级大小关系判断：

$$O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(2^n) < O(n!) \dots$$

CPU主频，一般都在 2~4GHz 左右，也就是说每秒钟包含数十亿个个时钟周期。由于执行一次运算操作需要花费数个时钟周期，我们在设计算法的时候需要保证运算次数不超过**每秒钟 10^8 次**。

空间按照题目给出的要求算一算就好，一般情况下数组开到几千万没有问题，记得**不要把数组开的太满**。





入门



题目描述

计算1-n的阶乘和对 $1e9+7$ 取余的结果.

即 $\sum_{i=1}^n i! \% 1e9 + 7$

$N \leq 1000000$



入门



题目描述

一共有 n 张地毯，编号从 1 到 n 。现在将这些地毯按照编号从小到大的顺序平行于坐标轴先后铺设，后铺的地毯覆盖在前面已经铺好的地毯之上。

地毯铺设完成后，Makik 想知道覆盖地面某个点的最上面的那张地毯的编号。

输入格式：

第一行，一个整数 n ，表示总共有 n 张地毯。

接下来的 n 行中，第 $i+1$ 行表示编号 i 的地毯的信息，包含四个正整数 a, b, g, k 每两个整数之间用一个空格隔开，分别表示铺设地毯的左下角的坐标 (a, b) 以及地毯在 x 轴和 y 轴方向的长度

第 $n+2$ 行包含两个正整数 x 和 y ，表示所求的地面的点的坐标 (x, y)

输出格式：

输出共 1 行，一个整数，表示所求的地毯的编号；若此处没有被地毯覆盖则输出 -1

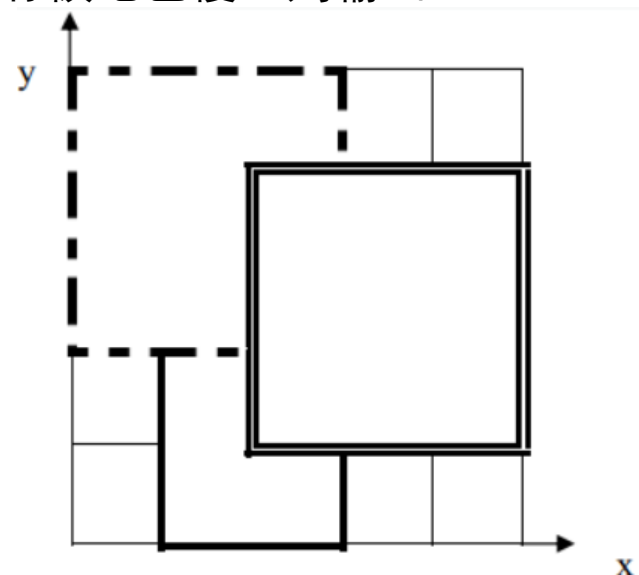
样例输入：

```
3
1 0 2 3
0 2 3 3
2 1 3 3
2 2
```

样例输出：

```
3
```

数据范围： $0 \leq n \leq 10,000, 0 \leq a, b, g, k \leq 100,000$



入门

一个想法：

模拟。

开一个二维数组，然后进行无脑填充。

```
for (int i = 1; i <= n; i++){
    cin >> a >> b >> g >> k;
    for (int dx = 0; dx <= g; dx++)
        for (int dy = 0; dy <= k; dy++)
            map[a + dx][b + dy] = i;
}
cin >> x >> y;
cout << map[x][y];
```

时间复杂度为 $O(n^2)$ 空间复杂度 $O(n^3)$

有点糟糕





入门



又有一个想法：
我们记录下每个地毯的信息，
对于(x,y)
我们在n个地毯中倒序查找
如果满足 $a \leq x \leq a+g$ && $b \leq y \leq b+k$
那么我们找到它了

时间复杂度： $O(n)$

空间复杂度： $O(n)$



2-

顺序查找

小蝌蚪见人便问：
你是我的麻麻吗

“





顺序查找



01:查找特定的值

总时间限制：1000ms 内存限制：65536kB

描述

在一个序列（下标从1开始）中查找一个给定的值，输出第一次出现的位置。

输入

第一行包含一个正整数 n ，表示序列中元素个数。 $1 \leq n \leq 10000$ 。

第二行包含 n 个整数，依次给出序列的每个元素，相邻两个整数之间用单个空格隔开。元素的绝对值不超过10000。

第三行包含一个整数 x ，为需要查找的特定值。 x 的绝对值不超过10000。

输出

若序列中存在 x ，输出 x 第一次出现的下标；否则输出-1。

样例输入

```
5
2 3 6 7 3
3
```

样例输出

```
2
```



2

枚举

小蝌蚪见人便问：
你是我的 Master 吗

“





枚举



定义：

又被称为穷举法。指从可能的集合中一一枚举各个元素，用题目给定的约束条件判定哪些是无用的，哪些是有用的。能使命题成立者，即为问题的解。

也就是对所有可能成为解的元素，挨个**批判一番**，检查它究竟是不是解。

最重要的就是确定**枚举对象**、**枚举范围**和**判定条件**。





枚举



【NOIP2001】一元三次方程

有形如： $ax^3 + bx^2 + cx + d = 0$ 这样的三个一元三次方程。给出该方程中各项的系数（ a 、 b 、 c 、 d 均为整数），并约定该方程存在三个不同实根（根的范围在-100至100之间），且根与根之差的绝对值不小于 1。

要求由小到大依次在同一行输出这三个实根（根与根之间留有空格），并精确到小数点后2位。





枚举



【NOIP2001】一元三次方程

首先想到的算法：一元三次方程的解的通式！

平时可以问谋财害命的网站，考试的时候压根记不住那种东西怎么办？

观察题目，发现答案范围又小，那就挨个试去呗...

```
for (int x = -100; x <= 100; x++)  
    if (a * x * x * x + b * x * x + c * x + d == 0)  
        cout << x;
```





枚举



【NOIP2001】一元三次方程

有形如： $ax^3 + bx^2 + cx + d = 0$ 这样的三个一元三次方程。给出该方程中各项的系数（ a 、 b 、 c 、 d 均为整数），并约定该方程存在三个不同实根（根的范围在-100至100之间），且根与根之差的绝对值不小于 1。

要求由小到大依次在同一行输出这三个实根（根与根之间留有空格），并精确到小数点后2位。





枚举



【NOIP2001】一元三次方程

首先想到的算法：一元三次方程的解的通式！

平时可以问谋财害命的网站，考试的时候压根记不住那种东西怎么办？

观察题目，发现答案范围还是那么小，那就挨个试去呗...

```
for (double x = -100; x <= 100; x += 0.01)
    if(a * x * x * x + b * x * x + c * x + d == 0)
        cout << x;
```



3

序列

这只是个游戏，
我原本是这样认为的。

“





子序列



- 给定 s 和一个长度为 n 的正整数序列。求出最短的子区间长度，满足这个子区间的和不小于 s .
- e.g. $n=5, s=11, a=(1,2,3,4,5)$.
- $(3,4,5)$ 的和为12，不小于11且最短。





子序列



- 如何暴力？
- 枚举左右端点。 $O(n^2)$.
- 如何优化？
- 二分长度。 $O(n \log n)$.





子序列



- 这题有 $O(n)$ 做法。
- 我们取两个指针，最开始都放在 a_1 位置。
- 这两个指针会夹住区间 $a_{[l,r]}$.我们现在可以通过移动这两个指针，来表示任意一段区间。
- 指针移动的时候，区间和的变化可以快速得到。





子序列



- 操作完了之后，现在我们把 l 固定在 a_1 ，要找到一个最短的区间，使得区间和 $\geq s$.
- 把右指针不停地往右移，直到 $a_{[l,r]}$ 的和 $\geq s$.
- 现在的 r 就是左端点为 a_1 时的最佳答案。
- 那么，如何算出其他左端点的最佳答案？





子序列



- 我们把 l 指针往右移一位，然后再去考虑移动 r .
- 由于 l 指针右移了一位，区间里少了一个数，所以区间和变小了。
- 因此， r 指针要么不移，要么只能往右移。我们继续往右移动 r 指针，直到新的区间和 $\geq s$ 为止。




```
scanf("%d%d",&n,&s);  
for(i=1;i<=n;i++)  
    scanf("%d",&a[i]);
```

```
l=1,r=1,su=a[1];
```

```
while(l<=n)
```

```
{
```

```
    while(su<s && r<=n)  
        r++,su+=a[r];
```

```
    if(su>=s) ans=min(ans,r-l+1);
```

```
    su-=a[l];
```

```
    l++;
```

```
}
```

```
printf("%d\n",ans);
```

C:\Users\Administrator\Desktop\讲课\code\sub.exe

```
5 11  
1 2 3 4 5  
3
```

```
-----  
Process exited after 2.227 seconds with return value  
请按任意键继续. . .
```



子序列

- 为什么我们要这样干呢？
- 因为，左指针和右指针都只会向右移动。
- 再怎么动也只能走过整个序列的长度！ $O(n)$.



陵墓设计

- 给定 n , 问将 n 拆成连续的数的平方和的方案。
- 输出这些方案。 $n \leq 10^{14}$.
- e.g. $2030 = 21^2 + 22^2 + 23^2 + 24^2 = 25^2 + 26^2 + 27^2$.



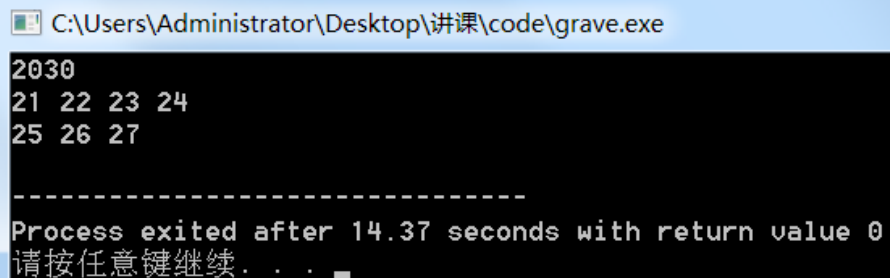
陵墓设计

- 既然 $n \leq 10^{14}$,我们只需要考虑 $\leq 10^7$ 的数的平方。
- 算出数组 a ,其中 $a[i] = i^2$.
- 现在我们需要在 a 上面取子区间, 使得区间和恰为 n .
- 把上面的代码稍微修改一下就行了。



陵墓设计

```
n=1000000;  
scanf("%d",&s);  
for(i=1;i<=n;i++)  
    a[i]=i*i;  
  
l=1,r=1,su=a[1];  
while(l<=n)  
{  
    while(su<s && r<=n)  
        r++,su+=a[r];  
  
    if(su==s)  
    {  
        for(i=l;i<=r;i++)  
            printf("%d ",i);  
        puts("");  
    }  
    su-=a[l];  
    l++;  
}
```



```
C:\Users\Administrator\Desktop\讲课\code\grave.exe  
2030  
21 22 23 24  
25 26 27  
-----  
Process exited after 14.37 seconds with return value 0  
请按任意键继续. . .
```



前缀和

$$\Delta a_i = a_i - a_{i-1}$$



前缀和

- 给定一个序列 a (初值全为0)。有很多次询问，每个询问形如：
- $A \mid r$ 询问将 $a_{[l,r]}$ 的区间和。
- 每次询问的复杂度要求 $O(1)$.



前缀和

- 我们搞出一个数组 s ,其中 $s_i = a_1 + a_2 + \cdots + a_i$.
- 那么： $a_l + a_{l+1} + \cdots + a_r = s_r - s_{l-1}$.
- 完事！



4

二分

易有太极，是生两仪
两仪生四象，四象生八卦

“





二分/三分



二分

适用范围：查找**单调函数（有序序列）**上的一个点（元素）

朴素查找时间复杂度 $O(N)$

二分查找时间复杂度 $O(\log N)$

重点：注意边界条件判断

三分

适用范围：查找**凸（凹）函数**的极值

两种写法：① 三等分 ② 不等分

例题：【[Luogu2571](#)】传送带





二分



```
int l = 1, r = n, ans = 0;
while (l <= r)
{
    int mid = (l + r) >> 1;
    if (a[mid] > x) r = mid - 1;
    else ans = mid, l = ans + 1;
}
```

<algorithm>

lower_bound(first, last, val, cmp)

upper_bound(first, last, val, cmp)





二分



应用一：二分查找

描述

在一个非降序列中，查找与给定值最接近的元素。

输入

第一行包含一个整数 n ，为非降序列长度。 $1 \leq n \leq 100000$ 。

第二行包含 n 个整数，为非降序列各元素。所有元素的大小均在 $0-1,000,000,000$ 之间。

第三行包含一个整数 m ，为要询问的给定值个数。 $1 \leq m \leq 10000$ 。

接下来 m 行，每行一个整数，为要询问最接近元素的给定值。所有给定值的大小均在 $0-1,000,000,000$ 之间。

输出

m 行，每行一个整数，为最接近相应给定值的元素值，保持输入顺序。

若有多个值满足条件，输出最小的一个。





二分



应用一：二分查找

描述

在一个非降序列中

输入

第一行包含一个整数

第二行包含n个整数

1,000,000,000

第三行包含一个整数

接下来m行，每行

定值的大小均在0

输出

m行，每行一个整数，为最接近相应给定值的元素值，保持输入顺序。

若有多个值满足条件，输出最小的一个。

```
while((l+1)<r)
{
    mid = (l + r) / 2;
    if(a[mid]<b)
        l = mid;
    else
        r = mid;
}
if((b-a[l])>(a[r]-b))
    cout << a[r] << endl;
else
    cout << a[l] << endl;
```

00。

在0-

10000。

所有给





二分答案



二分答案将最优性问题转化为可行性问题。

当问题直接求解较难时，二分答案可以为贪心等算法指明一个方向。

适用范围：答案具有单调性。答案为 x 时可行则答案小于等于 x 时都可行；答案为 x 时不可行则答案大于等于 x 时都不可行。

算法由二分查找和验证答案两个部分嵌套组成。二分时要清楚最终的答案是区间左端点，右端点还是中间点，如果没有把握则可以开一个新变量，每一次发现区间中点是可行解时都用中点更新。





二分答案



【NOIP2015】跳石头

一年一度的“跳石头”比赛又要开始了！

这项比赛将在一条笔直的河道中进行，河道中分布着一些巨大岩石。组委会已经选择好了两块岩石作为比赛起点和终点。在起点和终点之间，有 N 块岩石（不含起点和终点的岩石）。在比赛过程中，选手们将从起点出发，每一步跳向相邻的岩石，直至到达终点。

为了提高比赛难度，组委会计划移走一些岩石，使得选手们在比赛过程中的最短跳跃距离尽可能长。由于预算限制，组委会至多从起点和终点之间移走 M 块岩石（不能移走起点和终点的岩石）。





二分答案



【NOIP2015】跳石头

题目中要求的是**最短**跳跃距离的**最大值**，因此让我们来二分这个距离。检验的时候，只需要贪心地从起点开始，每一次都向前跳到不小于这个最小距离的最远的石头处。另外，如果最后一个跳上的石子离终点太近，还需要直接将其越过——上述过程中被越过的石子就是需要移除的石子，判断需要移除的石子数和 M 的大小关系即可判断出这个解是否可行。





二分答案



题目描述

Makik从前很苦恼他一直不长胡子。看到同学hth的胡子，Makik总是很无耻的偷笑... 有一天，然叔要带Makik参加n天的外出培训，在火车上，Makik突然发现自己长了胡子！然叔仔细观察了Makik的胡子，并且发现：1.胡子初始每天深夜都会长 v cm；2.每次在剃掉胡子之后胡子增长的速度会增加 s cm/天；Makik很伤心，并且由于来时并不需要剃须刀，所以只能借然叔的，他只允许Makik使用 x 次剃须刀，而且只允许在晚上睡前用。

输入格式

一行， n, s, v, x 四个整数

输出格式

输出在培训期间Franky的胡子最长的那天胡子的长度最短值

样例输入

6 1 1 2

样例输出

4

数据, $x, n \leq 100000, c, s \leq 10000$;



递归

“人类的本质是
 ‘人类的本质是
 复读机’
 ——hth”
 ——Makik





递归

A.复读机

B. + ①

C.人类的本质是什么?

D.人类的本质是什么?

A.复读机

B. + ①

C.人类的本质是什么?

D.人类的本质是什么?

A.复读机

B. + ①

C.人类的本质是什么?

D.人类的本质是什么?

A.复读机
B. + ①

人类的本质是复读机

```
void f(int pos){
    if (pos > 1000)
        return;
    printf("人类的本质是什么? \n");
    for (int i = 1; i <= pos + 5 * pos; i++) cout << " ";
    printf("A.复读机\n");
    for (int i = 1; i <= pos + 5 * pos; i++) cout << " ";
    printf("B.+①\n");
    for (int i = 1; i <= pos + 5 * pos; i++) cout << " ";
    printf("C.人类的本质是什么?\n");
    for (int i = 1; i <= pos + 5 * pos; i++) cout << " ";
    printf("D.");
    Sleep(800);
    f(pos + 1);
}
int main() {f(1);}
```





递归



递归的一般写法：

```
int f(int pos){  
    if(pos > MAX_Layer)  
        return Something;  
    doSomething();  
    f(pos + ?);  
    doSomething_again();  
    return SomethingEles;  
}
```





递归



描述

Makik为了减肥每天爬楼梯，减肥的同时还可以思考各种数学问题。

他可以每次走1级或者2级，输入楼梯的级数，求不同的走法数

例如：楼梯一共有3级，他可以每次都走一级，或者第一次走一级，第二次走两级也可以第一次走两级，第二次走一级，一共3种方法。

输入

输入包含若干行，每行包含一个正整数N，代表楼梯级数， $1 \leq N \leq 30$

输出

不同的走法数，每一行输入对应一行输出

样例输入

5
8
10

样例输出

8
34
89



递归

Makik为了减肥每天爬楼梯，减肥的同时还可以思考各种数学问题。
他可以每次走1级或者2级，输入楼梯的级数，求不同的走法数
例如：楼梯一共有3级，他可以每次都走一级，或者第一次走一级，第二次走两级
也可以第一次走两级，第二次走一级，一共3种方法。

思路一：模拟爬楼梯的路线
用递归的形式写出来长这样：

```
int f(int pos){
    if(pos >= n){
        if(pos == n) ans++;
        return;
    }
    f(pos + 1);
    f(pos + 2);
}
```

```
int f(int pos){
    if(pos > MAX_Layer)
        return Something;
    doSomething();
    f(pos + ?);
    doSomething_again();
    return SomethingEles;
}
```

时间复杂度： O （答案多大我有多大）





递归



输出1-n的全排列

$n \leq 8$

样例输入：

3

样例输出：

1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1





递归



输出1-n的全排列

一种思路：
循环嵌套

```
for (int a = 1; a <= 4; a ++){  
    for (int b = 1; b <= 4; b ++){  
        for (int c = 1; c <= 4; c ++){  
            for (int d = 1; d <= 4; d ++){  
                if(互不相同)  
                    printf("%d%d%d%d", a, b, c, d);  
            }  
        }  
    }  
}
```

但是你要写n层循环嵌套
怎么写n重循环？？





递归



输出1-n的全排列

递归：可以解决不定重循环的问题

```
void f(int pos){
    if(pos > n){
        print();
        return;
    }
    for (int i = 1; i <= n; i++){
        if(!use[i]){
            use[i] = 1;
            ans[pos] = i;
            f(pos + 1);
            use[i] = 0;
        }
    }
}
```

```
void print(){
    for (int i = 1; i <= n; i++){
        printf("%d ", ans[i]);
    }
}
```





递归



描述

Makik为了减肥每天爬楼梯，减肥的同时还可以思考各种数学问题。

他可以每次走1级或者2级，输入楼梯的级数，求不同的走法数

例如：楼梯一共有3级，他可以每次都走一级，或者第一次走一级，第二次走两级也可以第一次走两级，第二次走一级，一共3种方法。

输入

输入包含若干行，每行包含一个正整数N，代表楼梯级数， $1 \leq N \leq 30$

输出

不同的走法数，每一行输入对应一行输出

样例输入

5
8
10

样例输出

8
34
89





递归



时间上的浪费：

我们只需要有多少种方案，但是我们把所有方案都模拟了一边。

有没有一种方法让我们直接得到方案？

方法二：

$F[n] = f[n-1] + f[n-2];$





递归



注意

从前有座山山里有座庙庙里有个老和尚老和尚给小和尚讲故事
讲着讲着就圆寂了

因为没有边界条件



6

排序

一二三四五

上山打老鼠，老鼠打不着，烤个小竹鼠

“



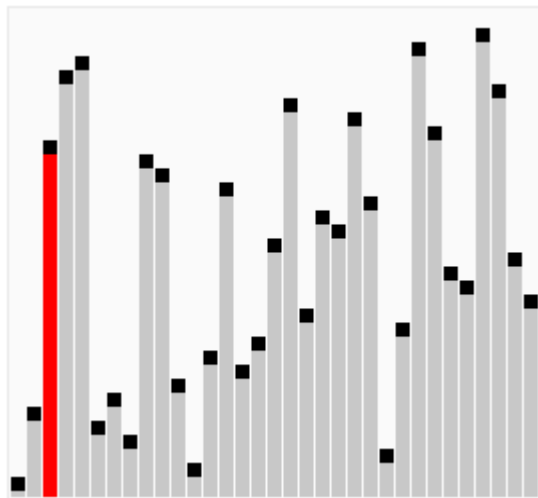


排序



算法复杂度分析：

冒泡算法 $O(n^2)$ 选择排序 $O(n^2)$



	8
	5
	2
	6
	9
	3
	1
	4
	0
	7





排序



有两个单调增的序列A与B，
A的长度为n， B的长度为m。
将其合并为一个单调增的序列C。

输入格式：

N, M

N个整数，表示序列A，保证递增

M个整数，表示序列B，保证递增

输出格式：

N+M个整数，用空格隔开

输入样例：

3 4

1 4 6

2 3 5 7

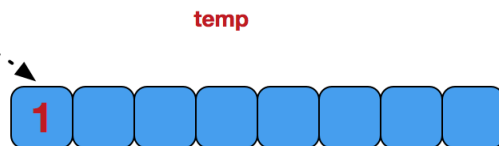
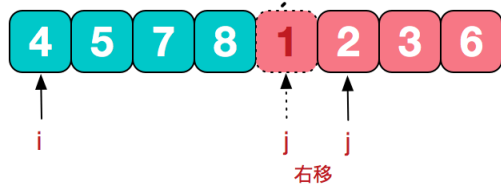
输出样例：

1 2 3 4 5 6 7

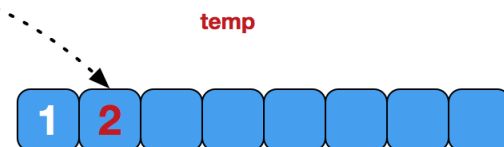
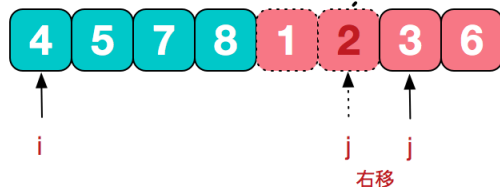


排序

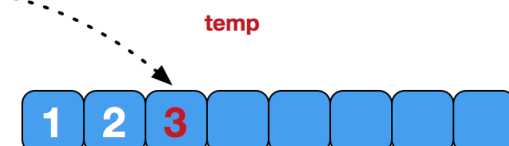
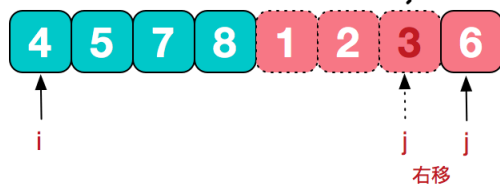
1<4, 将1填入temp数组, 右移j



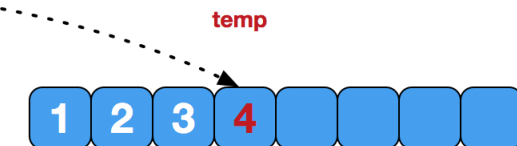
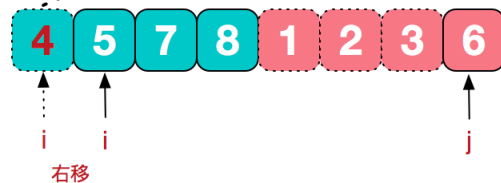
2<4, 将2继续填入temp数组, 右移j



3<4, 将3填入temp数组, 右移j



4<6, 此时将4填入temp数组, 右移i





排序



归并排序

把左边排好 把右边排好

把左边的左边排好，把左边的右边排好 把左边的右边排好，把右边的右边排好

.....

6 5 3 1 8 7 2 4



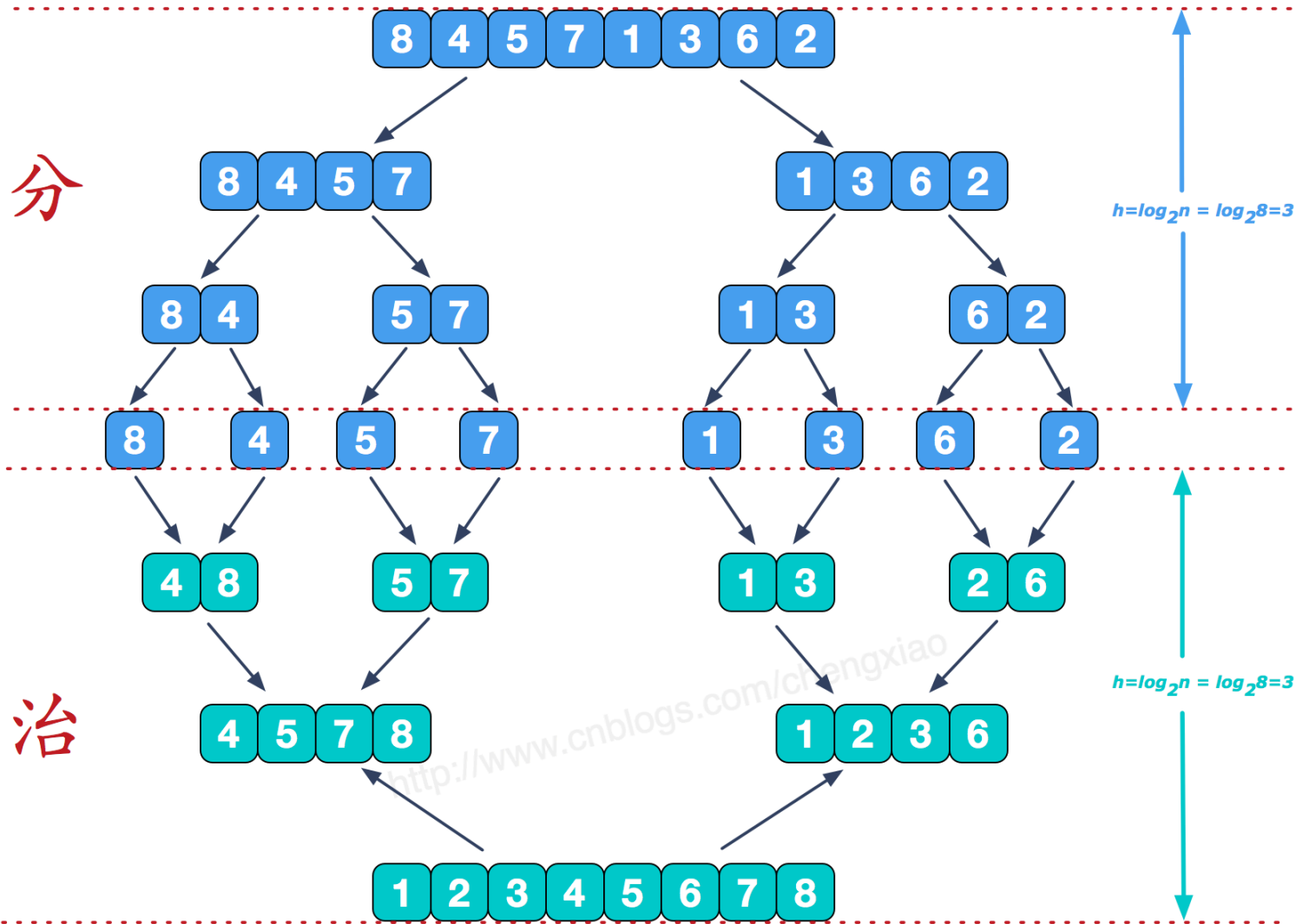


排序



分

治



排序

```
void mergeSortUp2Down(int* a, int start, int end){
    if(start >= end) return;
    int mid = (start + end)/2;
    mergeSortUp2Down(a, start, mid); // 递归排序a[start...mid]
    mergeSortUp2Down(a, mid+1, end); // 递归排序a[mid+1...end]
    // a[start...mid] 和 a[mid...end]是两个有序空间,
    // 将它们排序成一个有序空间a[start...end]
    merge(a, start, mid, end);
}

void merge(int* a, int start, int mid, int end)
{
    int tmp[end-start+1]; // tmp是汇总2个有序区的临时区域
    int i = start;        // 第1个有序区的索引
    int j = mid + 1;      // 第2个有序区的索引
    int k = 0;            // 临时区域的索引
    while(i <= mid && j <= end){
        if (a[i] <= a[j]) tmp[k++] = a[i++];
        else tmp[k++] = a[j++];
    }
    while(i <= mid) tmp[k++] = a[i++];
    while(j <= end) tmp[k++] = a[j++]; // 将排序后的元素, 全部都整合到数组a中。
    for (i = 0; i < k; i++) a[start + i] = tmp[i];
}
```

