算法基础——枚举和贪心

邓丝雨

牛客竞赛
AC.NOWCODER.COM

日期	时间	课程内容
5月14日	19：30——21：50	枚举（尺取法、前缀和、差分等）、贪心
5月18日		递归与分治思想（附各种排序和快速幂）、stl的运用
5月21日		二分、三分、01分数规划
5月25日	18：00——21：00	比赛一
5月28日	19：30——21：50	习题课1
6月1日		数据结构1：堆、并查集
6月4日		搜索1：深度优先搜索、广度优先搜索、搜索剪枝
6月8日		动态规划1：线性dp、背包问题、区间dp
6月11日	18：30——21：30	比赛二
6月15日	19：30——21：50	习题课2
6月18日		图论1：图论相关概念、最小生成树、最短路
6月22日		数学基础1：基本数论、组合数学（排列组合，容斥等）
6月25日		数据结构2：线段树、树状数组
6月29日	18：30——21：30	比赛三
7月2日	19：30——21：50	习题课3

课程说明与要求

- 希望大家能够养成写题解的习惯。
- 请大家独立完成编码。
- 知其然也要知其所以然，多问几个为什么。

一、什么是算法?

- 来考虑一个题目：《明明的随机数》 NC16669
- 明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N个1到1000之间的随机整数，对于其中重复的数字，只保留一个，把其余相同的数去掉。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作。

- 几种解法：
- 1.对数列进行去重
  ——没有标记的元素和他后面的元素两两比较，相同的则把后一个标记为不要，对去重之后的数组再排序（以冒泡为例）。
- 2.对数列进行排序（以冒泡为例），然后从小到大遍历，若当前a[i]和a[i-1]相等就不输出，否则就可以输出。
- 3.用一个数组b[i]表示i有没有出现过，每读入一个x，就将b[x]赋值为1，表示x出现过了，最后从0到1000遍历b数组，如果b[i] =1 就输出i。

一、什么是算法?

- 算法是解题过程的准确而完整的描述。它是一个有限规则的集合，这些规则确定了求解某一类问题的一个运算序列，对于某一类问题的任何初始输入，它能机械地一步一步地计算，并且通过有限步骤之后，计算终止并产生输出。

一、什么是算法?

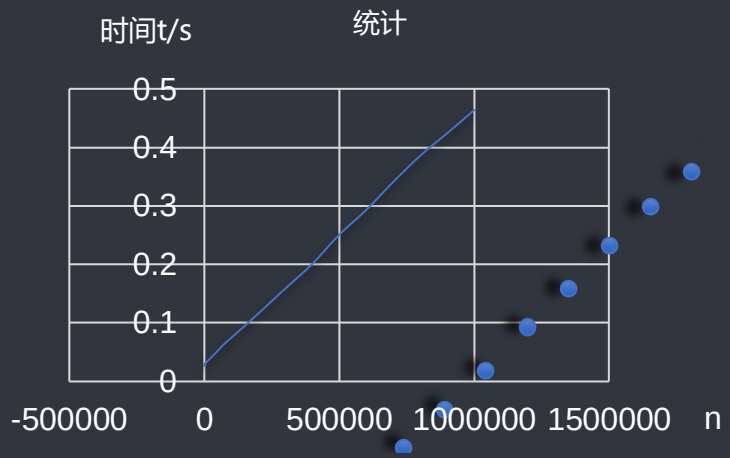
- 算法的特征:
- 1.有穷性：一个算法必须总是在执行有限步之后结束。
- 2.确定性：算法的每一个步骤必须是确切地定义的。
- 3.输入：一个算法 有0个或多个输入。
- 4.输出：一个算法有1个或多个输出。
- 5.可行性：算法中要执行的每一个计算步骤都是可以在有限时间内完成的.

二、怎样评价一个算法?

- 1.正确性
- 2.可读性
- 3.健壮性（容错性）——对不规范数据的处理能力（竞赛中一般不考虑）
- 4.时间复杂度
- 5.空间复杂度

三、时间复杂度

- 我们先来看“明明的随机数”一题两个算法的运行时间的比较
- 随着n的增大，算法的运行时间是怎么变化的?



三、时间复杂度

- 时间复杂度是衡量程序的运行速度的量度，它在衡量的时候忽略了硬件的差异。
- 它是一个定性描述程序运行时间和数据规模n的关系的函数，这个函数即程序执行基本操作（加减寻址赋值简单的数学函数等）的次数，记做 T(n)

```
for(int i = 1; i <= k; i++)
{
    ans = (ans + f[n][i]) % mod;
}
```

三、时间复杂度

- 大O表示法：考察n趋近于正无穷时的情况
- 当且仅当存在常数c>0和N>=1，
- 对一切n>N均有|g(n)|<=c|f(n)|成立
- 称函数g(n)以f(n)为渐进上界或者称g(n)受限于f(n)。记作g(n)=O(f(n))。

三、时间复杂度

- 实际计算时，往往直接算，忽略低次项和系数即可
- Eg:冒泡排序
- g(n)=(n-1)C+(n-2)C+(n-3)C+……+1C

  =1/2（n^2-n）C

  =O(n^2)

```
for (i=0; i<len-1; i++)
    for (j=0; j<len-1-i; j++)
        if (a[j] > a[j+1]) swap(a[j], a[j+1]);
```

三、时间复杂度

- 常见的时间复杂的
- O(1) 和输入数据规模无关
- O(logn) 一般我们默认底数是2，不是2也没关系，用换底公式之后就是常数了
- O($\sqrt{n}$)
- O(n) 线性时间复杂度
- O(n^2)
- O(n^3)……
- O(C^n) C是一个常数，指数级
- O(n!) 阶乘级

```cpp
#include<cstdio>
using namespace std;
int n,t;
int a[21];
int main(){
        scanf("%d",&n);
        a[0]=1;
        a[1]=3;
        for(int i=2;i<=n;i++)
        {
                t=0;
                for(int j=0;j<=i-2;j++)
                t=t+a[j];
                a[i]=a[i-1]+2*t+2;
        }
        printf("%d",a[n]);
        return 0;
}
```

```cpp
#include<cstdio>
using namespace std;
int n,m[21][21];
int main(){
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
                for(int j=i;j<=n;j++){
                        m[i][j]=i;
                        m[j][i]=i;
                }
        }
        for(int i=1;i<=n;i++){
                for(int j=1;j<=n;j++){
                        printf("%3d",m[i][j]);
                }
                printf("\n");
        }
}
```

```cpp
#include<cstdio>
using namespace std;
int n,m;
int main(){
        scanf("%d",&n);
        for(int i=2;i*i<=n;i++){
                if(n%i==0){
                        m=n/i;
                }
        }
        printf("%d",m);
        return 0;
}
```

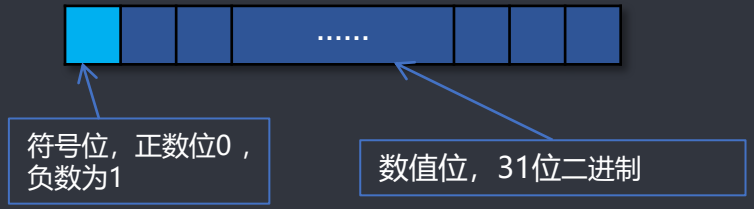
```cpp
#include<cstdio>
#include<cmath>
using namespace std;
int n;
double x,h;
int main(){
        scanf("%lf",&h);
        for(n=1;n<=9;n++){
                x+=h;
                h=0.5*h;
                x+=h;
        }
        x+=h;
        h=0.5*h;
        printf("%g\n%g",x,h);
        return 0;
}
```

四、空间复杂度

- 计算机内各种变量的存储

四、空间复杂度

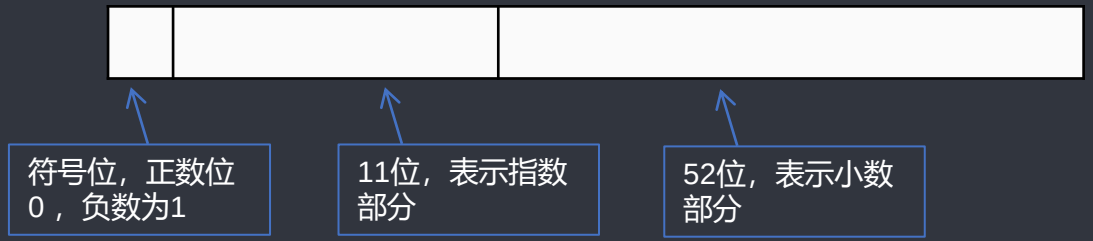
- int 32位二进制



- Long long 64位二进制
- Unsigned 用于修饰 int 或者long long， 将符号位变为数值位

四、空间复杂度

- Double 64位二进制 科学计数法存储 eg：2.33*10^2333



四、空间复杂度

- Char 8位二进制
- Bool 只有0和1 但还是8位二进制

四、空间复杂度

- 计算变量所占的空间
- 8位二进制 = 1个字节
- 1024字节 = lKB
- 1024kB = 1MB
- 1024MB = 1GB……
- 比赛时题目的空间限制位512M，在没有递归等其他消耗下，且只需要开一个int类型的数组，那么这个数组最大可以开到什么数量级?
- 一个长度为10^6的double类型数组占多少空间（说出数量级即可）？

枚举

- 什么是枚举

- ——列举
- 要点：不重复，不遗漏

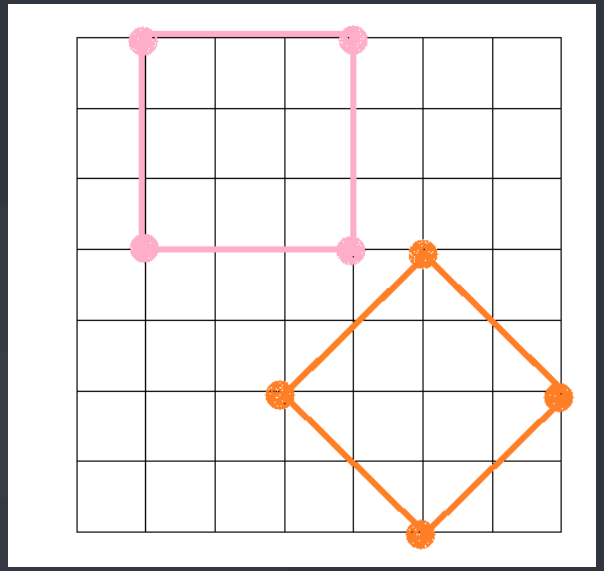
- 优化枚举的基本思路：——减少枚举次数
- 1.选择合适的枚举对象
- 2.选择合适的枚举方向——方便排除非法和不是最优的情况
- 3.选择合适的数据维护方法——转化问题

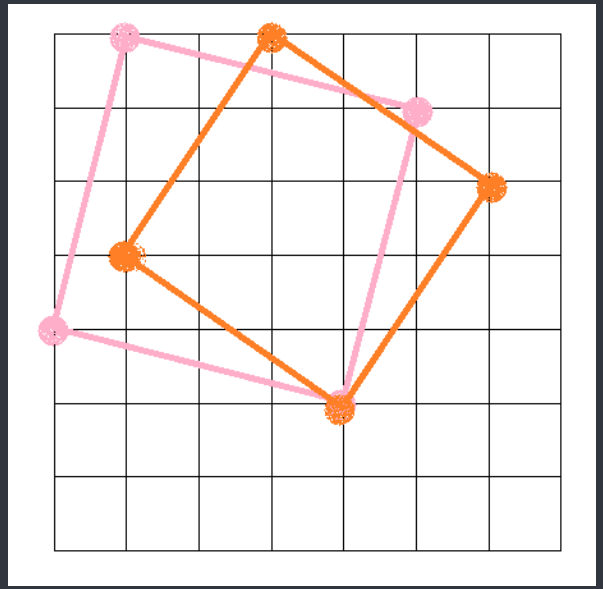
	A	B	C	D	E	F
1	学生信息表					
2	学号	班级	姓名	性别	出生年/月/日	家庭住址
3	001	111	xxx			

例1：最大正方形

- 在一个N*N(N<=100)矩阵中求一个最大的正方形使得该正方形的四个顶点都是有字符“#”构成。
- #*#***
- ******
- #*#*#*
- ******
- #*****
- ***#**

- 几个点能确定一个正方形?





例2：数列求和问题

- 给你一个数列{an}（1≤n≤100000），有q（1≤q≤100000）次询问，每次询问数列的第li个元素到第ri个元素的和。

例2：数列求和问题

- 主要的时间复杂度瓶颈在哪里?
- 对区间的查询需要将整个区间扫一遍

- 考虑如何进行转化
- ——将对区间的查询变为对区间端点的查询

例2：数列求和问题

- 我们可以用sum[i]存储前i个数的和，那么sum[i] = sum[i-1] + a[i]，当我们要查询第li个元素到第ri个元素的和时，用sum[ri] - sum[li-1]即可。
- 这样单次查询的复杂度是O(1)的，总复杂度是O(n+q)的。
- 这里的sum[i]是前缀和，前缀和也是算法竞赛当中非常常用的小技巧。

例3：数列修改问题

- 给你一个数列{an}（1≤n≤100000），有q（1≤q≤100000）次修改，每次把数列中的第li到第ri的每个元素都加上一个值ki，求所有的修改之后每个数的值。

例3：数列修改问题

- 主要的时间复杂度瓶颈在哪里?
- 对区间的修改需要将整个区间扫一遍
- 考虑如何进行转化
- ——将对区间的修改变为对区间端点的修改
- 考虑在区间加的过程中有什么值是在区间端点处发生了变化而区间内是没有变化的
- 是每个数与其前一个数的差值!

例3：数列修改问题

- 当我们对第li个到第ri个数加上ki时，第li个数与第li-1个数的差值增加了ki，第ri+1个数与第ri个数的差值减少了ki，而区间内部的相邻两个数的差值是不变的！
- 所以我们可以用数组delta[i]来维护第i个数和其前一个数的差值，（可以默认第一个数前面有一个0），然后当需要将[li,ri]区间的每一个数+ki时，只需要修改delta[li]和delta[ri+1]即可。
- 在所有的修改操作进行完之后，我们再对delta[i]求一次前缀和，就可以得到数列的每个元素的值了。
- 用数组delta[i]来维护第i个数和其前一个数的差值的办法叫做差分。

- 差分和前缀和是一对对称的操作（即对差分数组求前缀和就是原数组，对前缀和求差分也会得到原数组）

扩展：校门外的树 NC16649

- 某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数轴上的每个整数点，即0,1,2,...,L都种有一棵树。
- 由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树都移走后，马路上还有多少棵树。

扩展：校门外的树

- 原题数据范围：$1 \le L \le 10000$和$1 \le M \le 100$

扩展：校门外的树

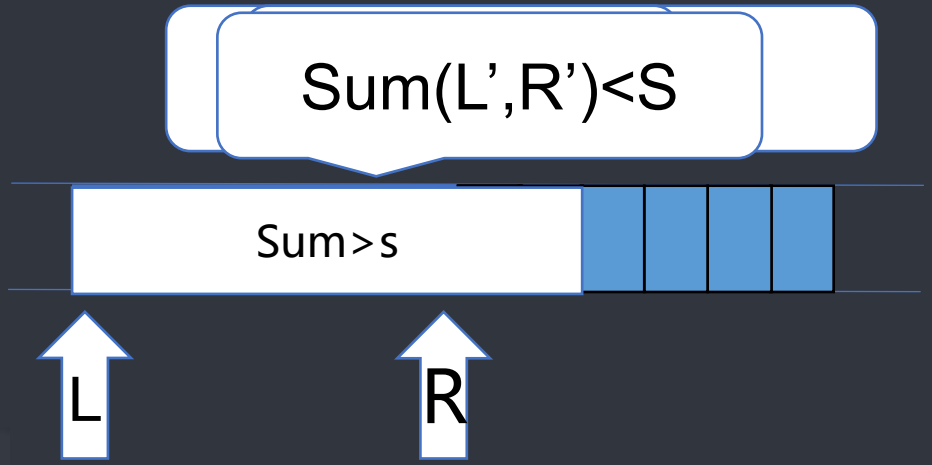
- 扩大数据范围：$1 \le L \le 100000$和$1 \le M \le 100000$

扩展：校门外的树

- 继续增大数据范围：1≤L≤10^9和1≤M≤100000

例4：

- 给定长度为n的整数数列以及整数S，求出总和不小于S的连续子串的长度的最小值，如果解不存在，输出0.



例5：

- 有一个N*M的灯泡，(N<=10,M<=100)，每次按某一个点可以使得其本身以及其上下左右共五个的灯的开关反向。给定初始状态（每个灯泡的亮或者灭），问：能否把所有灯都灭掉?

位运算介绍

- <<左移
- >>右移
- |或
- &与
- ~取反
- ^异或
- 请一定注意位运算优先级问题！！！请一定注意位运算优先级问题！！！请一定注意位运算优先级问题！！！

5	* / %	乘法/除法/取余
6	+ -	加号/减号
7	<< >>	位左移/位右移
8	< <=	小于/小于等于
	> >=	大于/大于等于
9	== !=	等于/不等于
10	&	按位与
11	^	按位异或
12	|	按位或
13	&&	与运算
14	||	或运算
15	?:	三目运算符

- a=a ^ b;
  b=a ^ b;
  a=a ^ b;
- 实现了什么功能?

位运算基础

- 去掉最后一位
- x >> 1
- 在最后加一个0
- x << 1
- 在最后加一个1
- (x<< 1)+1
- 把最后一位变成1
- x | 1
- 把最后一位变成0
- (x | 1) - 1
- 最后一位取反
- x ^ 1
- 把右数第k位变成1
- x | (1 << (k-1))
- 把右数第k位变成0
- x & (~ (1 << (k-1)))
- 右数第k位取反
- x ^ (1 < < (k-1))

贪心

什么是贪心?

- 贪心算法（又称贪婪算法）是指：在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，他所做出的是在某种意义上的局部最优解。
- 能够使用贪心算法的问题都是能严格证明贪心出的局部最优解就是所求的全局最优解的。

- 每次都选看起来最好的!

例1：排队接水

- 有n个人在一个水龙头前排队接水，假如每个人接水的时间为Ti，请编程找出这n个人排队的一种顺序，使得n个人的平均等待时间最小。
- n<=1000，ti<=1e6

例2：连数问题

- 设有n个正整数，将它们连接成一排，组成一个最大的多位整数。
- 例如：n=3时，3个整数13，312，343，连成的最大整数为34331213
- 又如：n=4时，4个整数7，13，4，246，连成的最大整数为7424613
- (n<=1000，每个数都在int范围内)

例3：区间覆盖（工作安排）

- 在0到L的数轴上有n个区间[li,ri]，现在需要你选出其中尽量多个区间，使得其两两不相交。（n<=100000）

区间编号	0	1	2	3	4	5	6	7	8	9	10	11
li	1	3	0	3	2	5	6	4	10	8	15	15
ri	3	4	7	8	9	10	12	14	15	18	19	20

例4：活动安排

- 给n个活动，每个活动需要一段时间Ci来完成，并且有一个截止时间Di，当完成时间ti大于截止时间完成时，会扣除ti-Di分，让你找出如何使自己所扣分的最大值最小。（n<=100000）

例5：国王游戏

- 恰逢 H 国国庆，国王邀请 n 位大臣来玩一个有奖游戏。首先，他让每个大臣在左、右手上面分别写下一个整数，国王自己也在左、右手上各写一个整数。然后，让这 n 位大臣排成一排，国王站在队伍的最前面。排好队后，所有的大臣都会获得国王奖赏的若干金币，每位大臣获得的金币数分别是：排在该大臣前面的所有人的左手上的数的乘积除以他自己右手上的数，然后向下取整得到的结果。

例5：国王游戏

- 国王不希望某一个大臣获得特别多的奖赏，所以他想请你帮他重新安排一下队伍的顺序，使得获得奖赏最多的大臣，所获奖赏尽可能的少。注意，国王的位置始终在队伍的最前面。
- 求获得奖赏最多的大臣获得的奖赏。
- $1 \le n \le 1,000$，$0 < a,b < 10000$。

- 课后习题：https://ac.nowcoder.com/acm/problem/collection/481