第十届蓝桥杯试题（QQ群文件中）

历年省赛试题汇总

第十届试题-答案

视频讲解

蓝桥杯官方训练系统

蓝桥杯部分VIP题目及省赛题目

NOIP历年普及组试题

骗分理论依据

- 蓝桥杯是一种 按测试数据给分的机制，不是只有正确和错误，而是 10%,20%,50%,80%,100% 这样来给分的 参考实验室OI Rank
- 样例是白送的答案
- 贪心算法是一种局部范围内近似的正解
- 只要时间给够，任何问题都可以 DFS
- 电脑不行的时候，交给人脑来算（数学技巧）

骗分技巧

- 题目中的样例有时候可能出现在评测数据中（直接特判输出样例中的答案即可）
- 针对数据范围比较小的情况，可以采取（手算一边直接放答案，DFS暴力一波，$random$ 一波）
- 针对答案范围固定的，比如答案只可能出现Yes 或者 No ， 0 或者 1， -1 ， $0 \leq ans \leq 10$ 等情况，直接$random$ 一波
- 猜想有时候也是一种答案
- 寻找规律，先手动输出大部分情况， 然后找出一种规律，直接套进去输出答案
- 贪心算法
- 暴力算法 DFS
- 利用系统工具，比如 EXCLE，MATLAB，Python，电脑日期等

1.无脑骗分

第八届蓝桥杯 第八题

第八题：包子凑数

本题总分：21分

小明几乎每天早晨都会在一家包子铺吃早餐。他发现这家包子铺有N种蒸笼，其中第i种蒸笼恰好能放Ai个包子。每种蒸笼都有非常多笼，可以认为是无限笼。

每当有顾客想买X个包子，卖包子的大叔就会迅速选出若干笼包子来，使得这若干笼中恰好一共有X个包子。比如一共有3种蒸笼，分别能放3、4和5个包子。当顾客想买11个包子时，大叔就会选2笼3个的再加1笼5个的（也可能选出1笼3个的再加2笼4个的）。

当然有时包子大叔无论如何也凑不出顾客想买的数量。比如一共有3种蒸笼，分别能放4、5和6个包子。而顾客想买7个包子时，大叔就凑不出来了。

小明想知道一共有多少种数目是包子大叔凑不出来的。

输入

第一行包含一个整数N。(1 <= N <= 100)
以下N行每行包含一个整数Ai。(1 <= Ai <= 100)

输出

一个整数代表答案。如果凑不出的数目有无限多个，输出INF。

例如，
输入：

```
2
4
5
```

程序应该输出：

```
6
```

再例如，
输入：

```
2
4
6
```

程序应该输出：

```
INF
```

样例解释：
对于样例1，凑不出的数目包括：1, 2, 3, 6, 7, 11。
对于样例2，所有奇数都凑不出来，所以有无限多个。

资源约定：
峰值内存消耗（含虚拟机） < 256M
CPU消耗 < 1000ms

骗分程序

```
#include <iostream>
#include <ctime>
using namespace std;
const int N = 110;
int a[N];
int main() {
    srand(time(NULL));
    int n;
    cin >> n;
```

```
    for(int i = 0;i < n; ++i) cin >> a[i];
    if(n == 2 && a[0] == 4 && a[1] == 5) {
        cout << 6 << endl;
    }
    else if(rand() % 2 == 0) cout << rand() % 10 << endl;
    else cout << "INF" << endl;
    return 0;
}
```

第九届蓝桥杯试题

第六题：递增三元组

【题目描述】

给定三个整数数组
A = [A1, A2, ... AN],
B = [B1, B2, ... BN],
C = [C1, C2, ... CN],
请你统计有多少个三元组(i, j, k) 满足：
1. 1 <= i, j, k <= N
2. Ai < Bj < Ck

【输入】

第一行包含一个整数N。
第二行包含N个整数A1, A2, ... AN。
第三行包含N个整数B1, B2, ... BN。
第四行包含N个整数C1, C2, ... CN。
1 <= N <= 100000 0 <= Ai, Bi, Ci <= 100000

【输出】

一个整数表示答案

【样例输入】

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

【样例输出】

```
27
```

骗分程序

```
#include <iostream>
#include <ctime>
using namespace std;
const int N = 1e5 + 10;
int a[N],b[N],c[N];
int main() {
    srand(time(NULL));
    int n;
    cin >> n;
    for(int i = 0;i < n; ++i) cin >> a[i];
    for(int i = 0;i < n; ++i) cin >> b[i];
```

```
    for(int i = 0;i < n; ++i) cin >> c[i];
    if(n == 3 && a[0] == 1 && a[1] == a[0] && a[2] == a[1]
    && b[0] == 2 && b[0] == b[1] && b[0] == b[2]
    && c[0] == 3 && c[1] == c[0] && c[2] == c[0]) {
        cout << 27 << endl;
    }
    else cout << 10 + (rand() % 100);
    return 0;
}
```

尝试骗分

求和

判断三角形

Powers Of Two

模拟与枚举

第十届蓝桥杯 第四题

第四题：数的分解

本题总分：10 分

【问题描述】

把 2019 分解成 3 个各不相同的正整数之和，并且要求每个正整数都不包含数字 2 和 4，一共有多少种不同的分解方法?

注意交换 3 个整数的顺序被视为同一种方法，例如 1000+1001+18 和 1001+1000+18 被视为同一种。

第六题：特别数的和

时间限制：1.0s内存限制：2560MB本题总分：15分
【问题描述】

小明对数位中含有 2、0、1、9 的数字很感兴趣（不包括前导 0），在 1 到 40 中这样的数包括 1、2、9、10 至 32、39 和 40，共 28 个，他们的和是 574。

请问，在 1 到 n 中，所有这样的数的和是多少?

【输入格式】

输入一行包含两个整数 n。

【输出格式】

输出一行，包含一个整数，表示满足条件的数的和。
【样例输入】

```
40
```

第八题：四平方和

四平方和定理，又称为拉格朗日定理：
每个正整数都可以表示为至多4个正整数的平方和。
如果把0包括进去，就正好可以表示为4个数的平方和。

比如：
5 = 0^2 + 0^2 + 1^2 + 2^2
7 = 1^2 + 1^2 + 1^2 + 2^2
（^符号表示乘方的意思）

对于一个给定的正整数，可能存在多种平方和的表示法。
要求你对4个数排序：
0 <= a <= b <= c <= d
并对所有的可能表示法按 a,b,c,d 为联合主键升序排列，最后输出第一个表示法

程序输入为一个正整数N (N<5000000)
要求输出4个非负整数，按从小到大排序，中间用空格分开

[CF672A Summer Camp]

[P5601 小D与笔试]

[Cut Ribbon]

[Flipping Game]

全排列

[全排列模板]

第七届 第三题

第三题：凑算式

凑算式

$$A + \frac{B}{C} + \frac{DEF}{GHI} = 10$$

△

这个算式中A~I代表1~9的数字，不同的字母代表不同的数字。

比如：
6+8/3+952/714 就是一种解法，
5+3/1+972/486 是另一种解法。

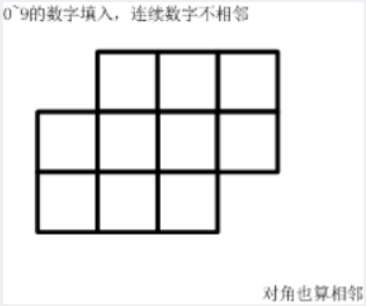
这个算式一共有多少种解法?

注意：你提交应该是个整数，不要填写任何多余的内容或说明性文字。

第七届第六题

第六题：方格填数

如下的10个格子



填入0~9的数字。要求：连续的两个数字不能相邻。
（左右、上下、对角都算相邻）

一共有多少种可能的填数方案?

请填写表示方案数目的整数。
注意：你提交的应该是一个整数，不要填写任何多余的内容或说明性文字。

位运算

第九届蓝桥杯

第二题：明码

【题目描述】

汉字的字形存在于字库中，即便在今天，16点阵的字库也仍然使用广泛。
16点阵的字库把每个汉字看成是16x16个像素信息。并把这些信息记录在字节中。
一个字节可以存储8位信息，用32个字节就可以存一个汉字的字形了。
把每个字节转为2进制表示，1表示墨迹，0表示底色。每行2个字节，
一共16行，布局是：
第1字节，第2字节
第3字节，第4字节
....
第31字节, 第32字节
这道题目是给你一段多个汉字组成的信息，每个汉字用32个字节表示，这里给出了字节作为有符号整数的值。
题目的要求隐藏在这些信息中。你的任务是复原这些汉字的字形，从中看出题目的要求，并根据要求填写答案。

```
###这段信息是（一共10个汉字）：
4 0 4 0 4 0 4 32 -1 -16 4 32 4 32 4 32 4 32 4 32 8 32 8 32 16 34 16 34 32 30 -64
0
16 64 16 64 34 68 127 126 66 -124 67 4 66 4 66 -124 126 100 66 36 66 4 66 4 66 4
126 4 66 40 0 16
4 0 4 0 4 0 4 32 -1 -16 4 32 4 32 4 32 4 32 4 32 8 32 8 32 16 34 16 34 32 30 -64
0
0 -128 64 -128 48 -128 17 8 1 -4 2 8 8 80 16 64 32 64 -32 64 32 -96 32 -96 33 16
34 8 36 14 40 4
4 0 3 0 1 0 0 4 -1 -2 4 0 4 16 7 -8 4 16 4 16 4 16 8 16 8 16 16 16 32 -96 64 64
16 64 20 72 62 -4 73 32 5 16 1 0 63 -8 1 0 -1 -2 0 64 0 80 63 -8 8 64 4 64 1 64
0 -128
0 16 63 -8 1 0 1 0 1 0 1 4 -1 -2 1 0 1 0 1 0 1 0 1 0 1 0 1 0 5 0 2 0
2 0 2 0 7 -16 8 32 24 64 37 -128 2 -128 12 -128 113 -4 2 8 12 16 18 32 33 -64 1
0 14 0 112 0
1 0 1 0 1 0 9 32 9 16 17 12 17 4 33 16 65 16 1 32 1 64 0 -128 1 0 2 0 12 0 112 0
0 0 0 0 7 -16 24 24 48 12 56 12 0 56 0 -32 0 -64 0 -128 0 0 0 0 1 -128 3 -64 1
-128 0 0
```

资料： 二进制

```cpp
#include <iostream>
using namespace std;

int main()
{
    freopen("in.txt", "r", stdin);
    int m, n, w[16];
    while (cin >> m >> n)
    {
        for (int i = 7; i >= 0; i--)
        {
            w[i] = m & 1;
            m >>= 1;
        }
        for (int i = 15; i >= 8; i--)
        {
            w[i] = n & 1;
            n >>= 1;
        }
        for (int i = 0; i <= 15; i++)
        {
            if (w[i] == 1)
                cout << '*';
            else
                cout << ' ';
        }
        cout << endl;
    }
}
```

二进制-子集枚举

算法训练 和为T

资料 子集枚举

STL

sort

map

deque

set

priority_queue

资料 STL

https://my.oschina.net/u/4353280/blog/3505566

树状数组(单点修改，区间查询)

问题引入 Acwing 1264

1264. 动态求连续区间和

题目 | 提交记录 | 讨论 | 题解 | 视频讲解

给定 n 个数组成的一个数列，规定有两种操作，一是修改某个元素，二是求子数列 $[a, b]$ 的连续和。

输入格式

第一行包含两个整数 n 和 m，分别表示数的个数和操作次数。

第二行包含 n 个整数，表示完整数列。

接下来 m 行，每行包含三个整数 k, a, b （$k = 0$，表示求子数列$[a, b]$的和；$k = 1$，表示第 a 个数加 b）。

数列从 1 开始计数。

输出格式

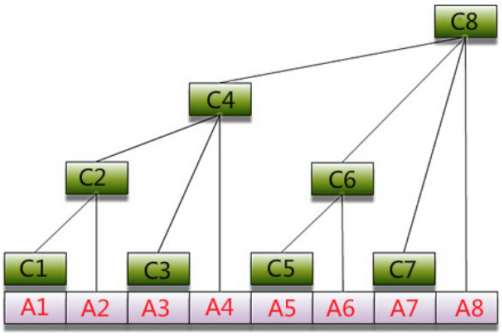
输出若干行数字，表示 $k = 0$ 时，对应的子数列 $[a, b]$ 的连续和。

数据范围

$1 \le n \le 100000$,
$1 \le m \le 100000$,
$1 \le a \le b \le n$

树状数组（Binary Indexed Tree(BIT), Fenwick Tree） 是一个查询和修改的复杂度都为 $log(n)$ 的数据结构。

观察下图：



令这棵树的结点编号为 $C_1, C_2, \ldots, C_n$。令每个结点的值为这棵树的值的总和，那么容易发现：

$$C_1 = A_1$$

$$C_2 = A_1 + A_2$$

$$C_3 = A_3$$

$$C_4 = A_1 + A_2 + A_3 + A_4$$

$$C_5 = A_5$$

$$C_6 = A_5 + A_6$$

$$C_7 = A_7$$

$$C_8 = A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8$$

$$\ldots$$

$$C_{16} = A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8 + A_9 + A_{10} + A_{11} + A_{12} + A_{13} + A_{14} + A_{15} + A_{16}$$

有一个有趣的性质：

设结点编号为 x，那么该结点区间为 2^k（其中 k 为 x 二进制末尾 0 的个数）个元素。因为这个区间最后一个元素必然为 A_x，

所以很明显，$C_n = A_{n-2^k+1} + ... + A_n$。

计算这个 2^k，也就是最低位的 1，可以这样写：

```
int lowbit(int x) {
    return x & (x ^ (x - 1));
}
```

利用机器补码特性，也可以写成：

```
int lowbit(int x) {
    return x & -x;
}
```

查询

当想要查询一个 $sum(1...n)$ 即 $(a_1 + a_2 + ... + a_n)$，可以依据如下算法即可：

step 1: 令 $sum = 0$，转第二步；

step 2: 假如 $n \leq 0$，算法结束，返回 sum 值，否则 $sum = sum + Cn$，转第三步；

step 3: 令 $n = n - lowbit(n)$，转第二步。

可以看出，这个算法就是将这一个个区间的和全部加起来，为什么是效率是 $log(n)$ 的呢?

证明：

$n = n - lowbit(n)$ 等价于将 n 的二进制的最后一个 1 减去。而 n 的二进制里最多有 $log(n)$ 个 1，所以查询效率是 $log(n)$ 的。

```
int getsum(int x) {
    int res = 0;
    for (; x; x -= x & (-x))
        res += t[x];
    return res;
}
```

修改

step 1：当 $i > n$ 时，算法结束，否则转第二步；

step 2：$Ci = Ci + x, i = i + lowbit(i)$ 转第一步。

$i = i + lowbit(i)$ 这个过程实际上也只是一个把末尾 1 补为 0 的过程。

修改一个节点，必须修改其所有祖先，最坏情况下为修改第一个元素，最多有 $log(n)$ 个祖先。

```
int change(int x) {
    for (; x <= maxn; x += x & (-x))
        t[x]++;
}
```

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
//tr[]数组 为树状数组
int a[N], tr[N];
int n, m;
//返回末尾最后一个1和后面的0
int lowbit(int x) {
    return x & (-x);
}
//树状数组的+操作 将x位置上的数+上y
void add(int x, int y) {
    for (int i = x; i <= n; i += lowbit(i)) {
        tr[i] += y;
    }
}
//求和以 1到x区间的和
int get_sum(int x) {
    int res = 0;
    for (int i = x; i > 0; i -= lowbit(i)) {
        res += tr[i];
    }
    return res;
```

```
}
int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        add(i, a[i]);
    }
    while (m--) {
        int k, x, y;
        cin >> k >> x >> y;
        if (k == 1) {
            add(x, y);
        }
        else if (k == 0) {
            cout << get_sum(y) - get_sum(x - 1) << endl;
        }
    }
    return 0;
}
```

例题 第五届蓝桥杯 小朋友排队

并查集

资料 并查集

第十一届蓝桥杯 网络分析