

$x_i, y_i \leq 10^9$ ，且保证所有给定的点互不重合。对于自由添加的整点，其横纵坐标不受限制。

测试点编号	$n \leq$	$k \leq$	$x_i, y_i \leq$
1~2	10	0	10
3~4		100	100
5~7	500	0	
8~10			
11~15		100	100
15~20			$10^9$

## CCF CSP-J/S 2022 第二轮入门级解题报告

### “乘方”解题报告

清华大学 张艺缤

#### 【概述】

本题是一道较为简单的题目，需要选手计算两个正整数的乘方，并对可能的溢出情况进行判断。主要考查简单的 C++ 程序设计能力、循环及分支判断等。

#### 【题目大意】

给定正整数  $a$  和  $b$ ，如果  $a^b$  不超过  $10^9$ ，就输出  $a^b$  的值，否则输出 -1。

数据范围： $a, b \leq 10^9$ 。

#### 【解法一】

$a$  的  $b$  次方即  $b$  个  $a$  相乘，通过循环语句逐次累乘即可。

注意到乘积的值可能很大，使用 `long long` 类型存储。计算完成后判断结果是否超过  $10^9$ 。

这样可以获得 60 分。

#### 【解法二】

在解法一中，由于  $b$  可能达到  $10^9$ ，直接循环可能会超时。

可以在每次累乘后判断乘积结果是否已经大于  $10^9$ ，如果是则可以直接退出循环并输出 -1。

另外，由于 1 的任意次方仍然是 1，因此当  $a=1$  时便可以无须计算，直接输出 1。

如此便可以获得 100 分，这是因为  $2^{30}$  就已经超过  $10^9$ ，因此循环累乘最多进行 30 次便一

定会退出。

### 【解法三】

实际上,可以直接使用 `cmath` 库中的 `pow(a, b)` 函数进行计算,并使用 `double` 类型存储计算结果。

这是因为 `double` 类型的精度有 52 位,当结果不超过  $10^9$  时, `double` 类型可以精确表达正整数而不会有精度误差;而当结果特别大时,计算结果会是 `inf`,与  $10^9$  比较大小也可以正常进行。

这样也可以获得 100 分。

### 【总结】

作为 CSP-J 的第一题,本题较为简单,选手只需掌握简单的分支、循环语句,并进行简单的分类讨论与特判,就可以通过此题。除此之外,本题还向选手普及了幂运算、`int` 类型的数据范围等知识,对于刚入门程序设计与算法竞赛的选手来说有一定的启发作用。

## “解密” 解题报告

清华大学 迟凯文

### 【概述】

本题以 RSA 算法这一密码学算法为背景(虽然题面中完全没有体现),主要考查内容是对 `if-else` 的熟练掌握、`long long` 类型的使用、C/C++ 中数学运算的使用,以及初中难度的数学知识。

### 【题目大意】

给定  $n, e, d$ , 求正整数  $p \leq q$ , 使得  $n = pq, ed = (p-1)(q-1) + 1$ 。

$k$  组数据。

$n \leq 1e18, ed \leq 1e18, k \leq 100\,000, m = n - de + 2 \leq 1e9$ 。

### 【解法一】

解方程:  $pq = n, p + q = m$  即可, 这等价于解  $p(m-p) = n$  这个一元二次方程。可以明确的是这是初中数学内容。

一元二次方程的求根公式里包含平方根,需要判断其是否为整数,可以先用 `sqrt` 求出实数解,随后在该解附近判断是否存在整数解(需要注意的是, `sqrt` 可能有一定的精度问题,不能直接判断 `sqrt()` 下取整是否是平方根,但是可以四舍五入)。也可以使用二分或分段打表等方法求解平方根。



**【解法二】**

注意到  $p(m-p)$  是以  $m/2$  为轴的二次函数，在轴两侧单调，因此可以二分出其等于  $n$  的位置。

**【总结】**

本题的思维难度较低，对实现细节有一定要求，包括判断一个数是不是完全平方数、判断一个数是否整除另一个数等。

**“逻辑表达式” 解题报告**

清华大学 张艺缤

**【概述】**

本题以 C++ 语言中的逻辑表达式和短路运算为背景，要求选手实现一个处理逻辑表达式的程序，并统计运算过程中的短路情况。本题难度并不高，在 CSP-J 系列赛事中属于中等，涉及栈、表达式树等数据结构。

**【题目大意】**

对于表达式  $a \& b$  而言，如果  $a$  为 0，则整个表达式的值一定为 0，如此便不需要再计算  $b$  的值；同理，对于表达式  $a \mid b$  而言，如果  $a$  为 1，则整个表达式的值一定为 1，如此也不需要再计算  $b$  的值。这两种情况统称为“短路”。

给定一个由 0, 1, &, |, (, ) 组成的逻辑表达式  $s$ ，规定运算顺序为“先算 & 再算 |，同一级别从左到右运算，有括号先算括号内”，需要计算出这个表达式的值，并统计计算过程中，两种“短路”各出现了多少次。

需要注意，“被短路”部分内部的“短路”不被统计在内，如表达式  $1 \mid (0 \& 1)$  中，内层的  $0 \& 1$  由于已被外层“短路”而不再计算，因此不被统计。

数据范围： $|s| \leq 10^6$ 。

**【解法一】**

对于串长不超过 5 的情况，可以直接手动枚举所有可能的输入并计算答案，可以获得 20 分。

**【解法二】**

对于一个表达式，我们尝试按照计算顺序进行计算，直到得到计算结果。在这个过程中，最后一步一定只剩下了一个形如  $a \& b$  或  $a \mid b$  的串，其中  $a$  和  $b$  分别是原串左、右部分的计算结果。



如此一来,以中间的运算符为根,表达式的左、右部分分别为左、右子树,再递归进行下去,便可以得到一个树形结构,这被称为表达式树。

建立表达式树的过程,可以每次遍历整个串,找到这个最中间的运算符是什么,然后直接向两侧递归。

当表达式树建好后,便可以 dfs 这棵表达式树,从下往上逐步计算出结果,并根据题意进行“短路”即可。

如此,建立表达式树的过程将会达到  $O(n^2)$  的复杂度,可以获得 50 分。

### 【解法三】

尝试优化上述建立表达式树的过程,可以使用栈来处理。

具体而言,由于原串是按照“左半部分——运算符——右半部分”的顺序给出的(这被称为“中序遍历”),我们便可以从前到后扫描整个串,并用栈记录当前遇到的、还没有被处理完的运算符。

每给出一个新的运算符时,便可以根据它与栈中运算符的运算顺序关系,推算出哪些运算符已经处理完毕,并进行弹栈和压栈操作。

这样可以将建立表达式树的复杂度降低至  $O(n)$ ,结合解法二的后半部分,可以获得 100 分。

### 【总结】

本题是一道较为经典的“表达式求值”模型,用栈来建立表达式树也是这类问题的常用思路。本题的主要创新点在于结合了 C++ 语言中的“短路”特性,需要选手在处理表达式时同时对“短路”情况加以识别和处理,具有很强的现实意义,可以很好地启发选手理解这一特性,并应用到实际代码编写中。总体而言,本题的难度不算高,CSP-J 组中水平较高的选手应当能正确处理并通过此题。

## “上升点列”解题报告

Google 苏黎世 钟诚

### 【概述】

动态规划是信息学竞赛中长盛不衰的考查点。本题是一道容易上手、模型比较典型的动态规划试题,因为它与同学们熟知的最长递增子序列问题有异曲同工之妙,故而很容易从中得到一些启发。而想要在这个升级版的问题中获得满分,又不失一定的思维深度,需要灵活地定义各点之间的“距离”,并在此基础上设计动态规划方程。

### 【题目大意】

给定  $n$  个整点,此外还可自由添加  $k$  个整点,从中选出若干个组成一个序列,要求任意



相邻两点的距离恰好为 1 且后一点位于前一点的右边或上边。求序列的最大长度。

### 【解法一】

当  $k=0$  时, 即没有自由添加的点时, 设  $f(x, y)$  表示点  $(x, y)$  为起点的最长连续上升点列的长度, 则

$$f(x, y) = \max \begin{cases} \text{后一点在右方时的最长长度}+1, \text{ 即} \begin{cases} f(x+1, y)+1 & // \text{若}(x+1, y) \text{ 有点} \\ 1 & // \text{若}(x+1, y) \text{ 无点} \end{cases} \\ \text{后一点在上方时的最长长度}+1, \text{ 即} \begin{cases} f(x, y+1)+1 & // \text{若}(x, y+1) \text{ 有点} \\ 1 & // \text{若}(x, y+1) \text{ 无点} \end{cases} \end{cases}$$

注意在用两重循环计算  $f(x, y)$  的值时, 要按一定的顺序: 先计算坐标较大的点, 再计算坐标较小的点, 这样才能保证计算时所需要用到的  $f$  值都已经算出来了。可以先按横坐标从大到小循环、再纵坐标从大到小循环, 也可以反过来。由于要遍历坐标轴里的所有点, 所以时间复杂度为  $O(\text{坐标轴里的点数})$ 。

### 【解法二】

在解法一的基础上, 如果  $k>0$ , 那么动态规划需要增加一个维度: 我们设  $f(x, y, z)$  表示点  $(x, y)$  为起点的最长连续上升点列的长度, 其中最多有  $z$  个自由添加的点, 则

$$f(x, y, z) = \max \begin{cases} \text{后一点在右方时的最长长度}+1, \text{ 即} \begin{cases} f(x+1, y, z)+1 & // \text{若}(x+1, y) \text{ 有点} \\ f(x+1, y, z-1)+1 & // \text{若}(x+1, y) \text{ 无点} \end{cases} \\ \text{后一点在上方时的最长长度}+1, \text{ 即} \begin{cases} f(x, y+1, z)+1 & // \text{若}(x, y+1) \text{ 有点} \\ f(x, y+1, z-1)+1 & // \text{若}(x, y+1) \text{ 无点} \end{cases} \end{cases}$$

由于要遍历坐标轴里的所有的点, 且要枚举所有可能的  $z$ , 所以时间复杂度为  $O(\text{坐标轴里的点数} * k)$ 。

### 【解法三】

考虑到坐标轴很大, 所以为了得到满分, 我们不能像解法二那样遍历坐标轴里所有的点。我们发现, 在最终的最长点列中, 如果只看最初给定的那些点, 那么相邻的两个点之间需要补上多少个自由添加的点是固定的, 例如  $(1, 1)$  和  $(3, 5)$  如果是相邻的, 那么它们之间要补上的一定是 4 个点, 而具体补的位置不影响点列的长度。

于是, 我们设  $f(i, z)$  表示第  $i$  个点为起点的最长连续上升点列的长度, 其中最多有  $z$  个后来补上的点。那么

$$f(i, z) = \max_{\text{所有的 } j, \text{ 使点 } j \text{ 在点 } i \text{ 的右上方}} f(j, z - \text{点 } i \text{ 和点 } j \text{ 之间需要补上的点数})$$

就这一转移方程, 有一个需要明确的问题: 我们并没有判断点  $i$  到点  $j$  之间的路线是否会途经最初给定的点。如果一定要计算, 是可以计算出来的, 但其实我们不需要去关心。理由很简单, 即使经过了部分最初给定的点, 那么我们在计算补上的点数时一定算多了, 因而它一定不是最优解。最优解一定是那些每次算补上的点数时, 都是实打实需要补上的情况, 而



上述方程也一定会包括这种情况。

由于动态规划的状态有  $nk$  个, 转移的复杂度为  $O(n)$ , 故这一解法总的时间复杂度为  $O(n^2k)$ 。

### 【总结】

本题题目浅显易懂, 考查了动态规划的经典模型。对动态规划熟练掌握的同学, 能通过一定的思考列出正确的转移方程。即使是动态规划的初学者, 也可以受最长上升子序列求法的启发, 列出一个基于枚举坐标轴中所有点的转移方程, 得到部分分。

## CCF CSP-J/S 2022 第二轮提高级试题

题目名称	假期计划	策略游戏	星战	数据传输
题目类型	传统型	传统型	传统型	传统型
目录	holiday	game	galaxy	transmit
可执行文件名	holiday	game	galaxy	transmit
输入文件名	holiday.in	game.in	galaxy.in	transmit.in
输出文件名	holiday.out	game.out	galaxy.out	transmit.out
每个测试点时限	2.0 秒	1.0 秒	2.0 秒	3.0 秒
内存限制	512 MB	512 MB	512 MB	1024 MB
测试点数目	20	20	20	25
测试点是否等分	是	是	是	是

### 提交源程序文件名

对于 C++ 语言	holiday.cpp	game.cpp	galaxy.cpp	transmit.cpp
-----------	-------------	----------	------------	--------------

### 编译选项

对于 C++ 语言	-O2 -std=c++14
-----------	----------------

### 注意事项 (请仔细阅读)

1. 文件名(程序名和输入输出文件名)必须使用英文小写。
2. C/C++ 中函数 `main()` 的返回值类型必须是 `int`, 程序正常结束时的返回值必须是 0。
3. 提交的程序代码文件的放置位置请参考各省的具体要求。
4. 因违反以上三点而出现的错误或问题, 申诉时一律不予受理。
5. 若无特殊说明, 结果的比较方式为全文比较(过滤行末空格及文末换行符)。
6. 选手提交的程序源文件必须不大于 100 KB。