用改进算法的思想解决规模维数增大的问题

广东韶关一中 张伟达

【关键字】 增大规模 改进算法 降维 分析 构造

【摘要】

我们常常会遇到一些特殊的问题，它们把我们能够解决的问题改了一改，增加了一维，或者增加了一个因素，从 1 到 2 或者是从 2 到 3，本文把它们统称规模维数增大的问题。

解决这类问题可以用改进算法的思想：本文第一部分先是概述这种思想；第二部分则从一道有趣的 IQ 题目说起，引入改进算法的基本思路和基本途径；第三部分则用多个例子解释说明如何改进算法；最后一部分是总结改进算法的思想。

【目录】

一、概述

每个学习信息学奥林匹克的选手总是要先学习一些基本的算法，然后才能把算法应用到题目当中去。但是，题目形式多种多样，这些算法往往是不能够直接套用的。有的时候我们分析到某个问题好像可以用某种方法解决，但是这个问题却与之前见过的问题有些不同，例如本文所讨论的情况：问题的规模维数比原问题大，一维的变二维的，二维的变三维的，等等。我们不妨把这类问题叫做规模维数增大的问题，解决这一类问题，往往需要观察问题的特殊性，改进原有算法，从而解决实际问题。

二、引子：从一道 IQ 题说起

【IQ 题题目】在走入正题之前，还是让我们来玩一道 IQ 题(据说是 IBM 公司招聘面试用过的题目)：有两根完全相同但分布不均匀的香（提示：不妨假设是蚊香，一圈圈的那种），每根香烧完的时间是一个小时，你能用什么方法来确定一段 45 分钟的时间？

【……】（思考中）

【思路】在公布答案之前，还是让我们来循序渐进地思考下去吧。

要确定 45 分钟的时间，而每根香烧完的时间是一个小时，现在应该有三个模型让我们思考：

A.不减小每根香的计时，两根香加在一起计时是 45 分钟；

B.只用一根香，使其计时减小，直接计时 45 分钟；

C.把两根香的计时都减小，使两根香加起来是 45 分钟。

A 模型显然是不可能成立的；B、C 模型的共同点是：我们必须使一根香的计时减小。但是怎样减，能减成什么呢？显然是不能直接把它们分开的，因为香的分布不均匀。由于这里笔者已经降低了难度，根据提示，比较容易想到香是可以两头一起烧的。这样，我们就能把一根香的计时减成半个小时。方案已经更进一步了。通过这一步，B 模型只用一根香，是很容易被排除的。

余下的选择只有“C 模型+两头烧方法”了，但是 C 模型不能直接应用“两头烧”，因为这样套用的话，0.5+0.5 仍然等于 1。我们需要对“两头烧”做进一步改进：如果一根香只烧了一头，当剩余时间为 t 的时候，我们把另一头也点燃——那么，我们就能够计出 t/2 的时间了（前提是我们已知当前剩余时间 t）。特别地，当 t=60min 时，t/2=30min。

根据这一改进后的方法，我们很容易就得出正确的解法：分别点燃第一根的两头和第二根的一头，第一根烧完的时候，已经过了 30 分钟；第二根还剩 30 分钟，点燃第二根的另一头；当第二根也烧完了，即时间又过了 15 分钟。那么我们计出的总的时间就为 45 分钟了。

【小结】上面这个例子中，我们从构造模型和改进算法两方面入手，一步一步地达到了优化的解法。主要流程是：

1. 找出原始解法和可能改进的方向（即分析成 A、B、C 模型）；
2. 分析算法的原理（由烧一根香计时半小时，引申为烧剩 *t* 的时候点两

头就能计时 $\frac{t}{2}$）；

3. 改进算法（改进的过程中，往往不是依靠算法改进算法本身，反而是利用算法的内涵、实质，结合问题，构造算法）；
4. 解决问题（我们得出了正确的解法）。

但是如何来改进原有的算法呢，笔者认为可以有以下途径：

1. 直接增加算法的规模，解决问题
2. 用枚举处理增加的规模，从而解决问题
3. 用贪心解决增加的规模，从而解决问题
4. 以上各个途径的综合运用

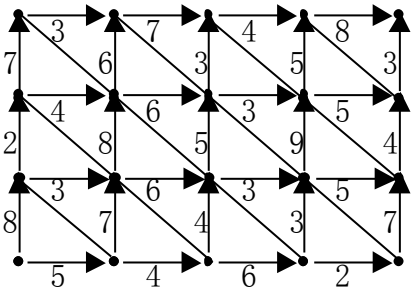
围绕这一主线，笔者将用例题做进一步的阐明。

三、改进算法的途径

(1)直接增加算法的规模，解决问题

解决简单的问题，常常可以采用直接处理的方法，但是很多时候还是需要一些技巧。

【例一】街道问题及其扩展。（经典问题）



【题目大意】〖原题〗在右图中找出从左下角到右上角的最短路径，每步只能向右方或上方走。〖扩展〗在地图中找出从左下角到右上角的两条路径，两条路径中的任何一条边都不能重叠，并且要求两条路径的总长度最短。

【问题分析】原题是一道简单而又典型的动态规划题，很显然，可以用这样的动态规划方程解题：$f_{i,j}=\min\begin{cases}f_{i-1,j}+\text{Distance}_{(i-1,j),(i,j)}\\ f_{i,j+1}+\text{Distance}_{(i,j+1),(i,j)}\end{cases}$（这里 Distance 表示相邻两点间的边长）

但是对于扩展后的题目：再用这种"简单"的方法就不太好办了。如果非得套用这种方法的话，则最优指标函数就需要有四维的下标，并且难以处理两条路径“不能重叠”的问题。

我们来分析一下原方法的实质：按照图中的斜线来划分阶段，即阶段变量 k 表示走过的步数，而状态变量 x_k 表示当前处于这一阶段上的哪一点。这时的模型实际上已经转化成了一个特殊的多段图。用决策变量 u_k=0 表示向右走，u_k=1 表示向上走，则状态转移方程为：$x_{k+1}=\begin{cases}x_k+1-u_k, k\le row\\ x_k-u_k, k>row\end{cases}$(这里的 *row* 是地图竖直方向的行数)。通过这一步观察，我们就会发现这个问题很好解决，只需要

加一维状态变量就成了。即用 $X_k=(a_k,b_k)$ 分别表示两条路径走到阶段 k 时所处的位置，相应的，决策变量也增加一维，用 $u_k=(x_k,y_k)$ 分别表示两条路径的行走方向。状态转移时将两条路径分别考虑：

$$\begin{cases}a_{k+1}=a_k+1-x_k,b_{k+1}=b_k+1-y_k,k\le row\\ a_{k+1}=a_k+x_k,b_k+1=b_k+y_k,k>row\end{cases}$$

【小结】从这个例子可以看出，改进的时候不能只依靠原始算法，还要分析原始算法的本质。

(2)用枚举处理增加的规模，从而解决问题

【例二】旅行（广东省奥林匹克竞赛 2001）

【题目大意】给出一个 N×M 的数字矩阵，要求这个矩阵的一个子矩阵，并要求这个子矩阵的数字和最大。

【问题分析】初一看题目，想到枚举每一个子矩阵，求出子矩阵的和，比较得出最大值。这样，时间复杂度达到 $O(N^4)$，显然不可以接受。因为它是两维的问题，我们可以尝试着把维数降低。

先看一维时候的情况：在数列 a_i 中找出和最大的一段。

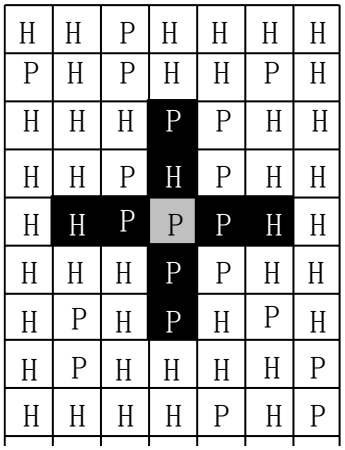
对于一维的子问题，可以这样来想：如果用最基本的方法，可以枚举每个子序列。但是如果纯粹三重循环，枚举头，枚举尾，枚举中间求和，显然是太浪费了。其实我们计算 $S_{i,j}$ ($S_{i,j}=a_j+a_{j+1}+\cdots+a_i, j<i$)时，可以由 $S_{i,j}=S_{i-1,j}+a_i$ 获得。这样，动态规划的思路就明显了，$S_{i,j}$ 与 $S_{i-1,j}$ 有直接的关系，而且我们还可以发现 $S_{i,*}$（表示从 $S_{i,1}$ 到 $S_{i,i-1}$ 构成的数列）只比 $S_{i-1,*}$ 多了一项。于是简单的动态规划就出来了，用 f_i 表示以 i 为终点的数列的最大和，有：

$$f_i=\begin{cases}\max(f_{i-1}+a_i,a_i),i>1\\ a_1,i=1\end{cases}$$

不过，我们得到的只是子问题的算法，如何改进算法能使它适用于矩阵的问题呢？我们需要找出矩阵和数列的关系：矩阵是若干条数列，但是它又不仅仅是简单的若干条数列，不同数列中的相同位置的数也构成了联系。我们可以尝试着构造上下、左右同时进行的动态规划，但是也许我们无能为力。对于这道题的数据规模，似乎没有必要构造 $O(N^2)$时间的算法。不难想到，枚举上下方向的数列和，在左右方向用动态规划求解，可以得到时间复杂度在 $O(N^3)$的算法。

【小结】在这个例题中，我们用了最简单的方法——枚举，来改进一维数列的最大子序列的解法，从而解决问题的规模的增加。但是枚举这个简单的方法，却能解决很复杂的问题。

【例三】炮兵阵地。（NOI2001）



【题目大意】在图中标为 P 的格子中放炮兵，如图灰色区域放了炮兵后，黑色区域不能放炮兵，问图中最多能放多少炮兵？

【问题分析】简化的问题：若只有一列，问最多可以放多少炮兵。

可以用贪心，尽量把炮兵往上放，也可以这样递推：用 f_i 表示前 i 行最多能放炮兵数，a_i 表示第 i 行的地形，则边界条件：$f_1=\begin{cases}1, a_1='P'\\ 0, a_1='H'\end{cases}$，动态规划方程：

$$f_i=\max\begin{cases}f_{i-3}+1, i>3且a_i='P'\\ 1, 1<i\le 3且a_i='P'\\ f_{i-1}, i>1\end{cases}$$

但是当问题扩展到多列的情况，就复杂了：不仅要考虑一列是否能引用上一行的函数值，而且要多列同时考虑。但是容易观察到，m<<n，最大只有 10，又因为炮兵攻击范围很大，实际上能放炮兵的组合数很小，可以考虑枚举组合，从而得到改进的动态规划方法：以两行为一个状态，s_1 表示状态中上行的炮兵布置情况，s_2 表示下行的布置情况，（可以预先枚举所有可能的情况并给这些情况标号）

当第 i 行能不能布置 s_1 或第 i+1 行不能布置 s_2 时，或者 s_1,s_2 在同一位置布置了炮兵时，都是不符合情况的，令此时 $f_i(s_1,s_2)=0$；

否则：$f_i(s_1,s_2)=\max(f_{i-1}(s,s_1))+\text{count}\,s_2$，其中 counts_2 表示 s_2 的炮兵个数；s 表示该 s_1,s_2 的前一行的炮兵布置情况，而且满足：不存在这样的 $j(j\le m)\&(s_2[j]=s[j]=1)$。

这样到最后的最优解就是 $\max(f_{n-1}(s_1,s_2))$

最终复杂度分析，时间复杂度大约 $O(R^3n)$（R 是一行状态的组合数）

(3)用贪心解决增加的规模，从而解决问题

【例四】求网络的最小费用最大流。（经典问题）

显然，求最小费用最大流可以由求最大流的算法改进。求网络的最大流，简单地说来，是每次寻找一条连接源点和汇点的可增广路径并由之扩充网络流，直到不存在这样可增广路径，则得出最大流。

那么，又如何把费用也考虑进去呢？我们先来看看网络的费用的定义吧。

网络的费用一般是指在每一段弧上弧的费用与弧的流量的乘积的和。（弧的费用由可能为负）所以网络的最小费用最大流是指可行流中费用最小的。不难看出，可行流每增加 1，所增加的费用都应该是最小的（事实上应该是减小得最多的）。这样可以得出一个改进：每次选取一条费用最小（而且非正）的可增广路径，直到最终不存在费用非正的可增广路径。这样用贪心的策略就能解决问题了。

(4)多种途径的综合运用

【例五】Team Selection (Balkan OI 2004 Day1)

【题目大意】IOI 要来了，BP 队要选择最好的选手去参加。幸运地，教练可以从 N 个非常棒的选手中选择队员，这些选手被标上 1 到 N(3 ≤ N ≤ 500000)。为了选出的选手是最好的，教练组织了三次竞赛并给出每次竞赛排名。每个选手都参加了每次竞赛并且每次竞赛都没有并列的。当 A 在所有竞赛中名次都比 B 前，我们就说 A 是比 B better。如果没有人比 A better，我们就说 A 是 excellent。求 excellent 选手的个数。

如数据：

```
10
2  5  3  8  10 7  1  6  9  4
1  2  3  4  5  6  7  8  9  10
3  8  7  10 5  4  1  2  6  9
```

则 excellent 选手是 1，2，3，5。

【原始思路】一拿到这一题，很容易会有以下的思路。

〖原始算法〗第一眼看到这道题往往想到枚举。简单地根据 better 和 excellent 的定义，现只需要枚举每一个选手 X，判断 X 是否 excellent。这可以通过另一重循环，枚举另一选手 Y，判断 Y 是否比 X better。判断是容易的，我们只需要简单地判断 X 和 Y 的三次排名。因此这一算法的时间复杂度是 $O(N^2)$，对于这道题，不能满足要求。

主要流程 1：

```
for(X 从 1 到 N)
    for(Y 从 1 到 N)
        判断 Y 是否比 X better
```

〖改进一〗原始算法是可以改进的。不难看出，如果让 X 依照第一次竞赛的名次循环，枚举 Y 时只需要枚举在第一次竞赛中排在 X 前面选手即可，因为第一次竞赛排在 X 后的选手一定不可能比 X better。不过这样小的改进提起来只是开阔开阔思路，它对时间复杂度不会有太大影响。

主要流程 2：

```
for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名)
    for(Y 从第一次竞赛的第 1 名到第一次竞赛的第(X-1)名)
        判断 Y 是否比 X better
```

〖改进二〗如果我们进一步观察，我们能优化上述算法的时间复杂度。我们发现：如果 Y 不是 excellent（因为有 Z 比 Y better），当我们检查 X 是否 excellent 时，我们检查了 Z 是否比 X better 的话，可以不检查 Y。因为如果 Y 比 X better，

那么 Z 一定比 X better。然后可以通过简单地修改“改进一”。在判断 X 是否 excellent 的时候，我们只需要在当前的 excellent 选手集合中取得 Y 即可。

主要流程 3：

for(X 从第一次竞赛的第 1 名到第一次竞赛的第 N 名)

for(Y 枚举当前已知的 excellent)

判断 Y 是否比 X better

这样，我们能把时间复杂度减少到 O(NK)（设 K 是 excellent 选手的个数）。由于这题 K 可能很大，算法效果仍然不是很理想。

【原始思路小结】这里的原始算法是直接根据原始模型模拟出来的，改进一和改进二都单纯地根据原始算法的设计缺陷来“改进”（这个改进没有利用问题的特殊性，不是本文所要阐述的“改进”），所以最后的时间复杂度没有质的进展。

【降维思路】本道题规模还是很明显的从二维扩充到三维。所以还是先用降维的思想，我们先解决选手之间只进行两次竞赛的问题。

两次竞赛的问题同样可以套用改进二的普遍算法。为了方便说明，我们设第一次竞赛排名依次为 A_i（表示第一次竞赛的第 i 名是 A_i），A_i 号选手在第二次竞赛中的排名的为 $B[A_i]$（注意 $B[A_i]$与 A_i 的含义不同）。参考“主要流程 3”：先是 $X=A_1$，显然 A_1 是 excellent；接着 $X=A_2$，只需要比较 $B[A_1]$与 $B[A_2]$大小，不妨假设 $B[A_2]<B[A_1]$，则 A_2 也是 excellent；接下来，$X=A_3$，我们需要比较 $B[A_1]$与 $B[A_3]$，$B[A_2]$与 $B[A_3]$……假设 A_3 也是 excellent；$X=A_4$ 时，我们需要比较 $B[A_1]$与 $B[A_4]$，$B[A_2]$与 $B[A_4]$，$B[A_3]$与 $B[A_4]$，假设 A_4 也是 excellent；……。也许这里有心人会发现什么，对了，比较 $B[A_1]$与 $B[X]$，$B[A_2]$与 $B[X]$，$B[A_3]$与 $B[A_i]$……的时候，只要有任何一个 $B[A_j]<B[A_i](j<i)$，A_i 就不是 excellent 的，那么，我们只需要用 $\min(B[A_j])$与 $B[A_i]$比较！

〖改进三〗（适用于两次竞赛）以上我们看到了一个特殊的例子：两次排名完全相反，但是我们可以从中提炼出可行的改进算法：对于两次竞赛的情况，当 $X=A_i$ 时，设 $best_i = \min(B[A_j])(j < i)$，则 $B[A_i]$只需要与 $best_i$ 比较即可。时间复杂度竟然达到 O(N)。

【降维思路小结】在这次分析中，我们从两次竞赛——简化后的问题出发，通过简单的观察和思考，得出了改进的算法，但是优化后的算法要应用到三维的情况还有很长的路要走。

【扩展思路】我们怎样把“改进三”应用到三维的情况从而解决问题呢？

改进三的思路能否再明确些？改进三的本质是什么？其实，X 依照第一次竞赛的名次循环，$X=A_i$ 时，因为 $best_i = \min(B[A_j])(j < i)$ 中，$j<i$ 这样就保证了 A_j 在第一次竞赛中名次一定比 A_i 前；如果 $best_i<B[A_i]$，这样就保证了 A_j 在第二次竞赛中名次比 X 前；总之则必然有 A_j 比 A_i better。

〖改进四〗能否把改进三扩展到三次竞赛上面去呢？为了方便说明，我们还需要设第三次竞赛中 A_i 选手名次为 $C[A_i]$。我们假设这样一个过程：

(1)当 $X=A_i$，设 $j<i$，则可以保证 A_j 在第一次竞赛中名次比 A_i 前，若不存在这样的 j，则 A_i 是 excellent；

(2)在符合(1)条件的所有 A_j 中找出能满足 $B[A_j]<B[A_i]$的选手，保证 A_j 在第二次竞赛中名次比 A_i 前，若不存在这样的 j，则 A_i 是 excellent；

(3)在符合(1)(2)条件的所有 A_j 中找出 min(C[A_j])，比较 min(C[A_j])与 C[A_i]，若 min(C[A_j])<C[A_i]，则保证 A_j 在第三次竞赛中名次比 A_i 前，则必有 A_j 比 A_i better，A_i 必不是 excellent；反之 A_i 是 excellent。

以上三个条件即为："A_i 是 excellent"等价于"不存在这样的 $j \in \{j \mid j < i, B[A_j] < B[A_i]\}$，使 $\min(C[A_j]) < C[A_i]$"。

〖改进五〗显然，如果我们仍然用枚举的方法来实现第(2)步，则最终的算法复杂度仍然是 $O(N^2)$。为什么我们却说这是改进过后的算法呢？因为我们实际上利用问题的特殊性，改进二是纯粹的枚举，进行的第(1)步以后要第二第三次竞赛同时比较，实现起来是很难优化的。但是改进四则体现了第二第三次竞赛分步选取的思路，利用这一点能否在实现中改进算法的复杂度呢？

我们可以比较，改进二和改进四是不同的，在进行了第(1)步以后，改进二需要同时考虑第二第三次竞赛成绩，两次成绩同时考虑并不是严格有序的，有时候我们不能比较二元组(B[A_i],C[A_i])与(B[A_j],C[A_j])的大小（例如当 B[A_i]<B[A_j]，但 C[A_i]>C[A_j]时）。

而改进四的第(2)步可以先考虑第二次竞赛的成绩，如果我们把第二次竞赛的成绩用有序数列来记录（设这个序列为 D_k，即把 B[A_j](j<i)从小到大排序），我们能轻易找到符合条件的 A_j。

〖改进六〗由改进五可以知道，我们单单从改进第(2)步，时间复杂度仍然不理想的，能否切实把第(2)步和第(3)步结合起来？经过了改进五的改进，我们把查找到的符合 B[A_j]<B[A_i](j<i)的 A_j 在一个有序数列中记录下来，这样，符合条件的 A_j 必然在数列中是连续的，设它是从 D_1 到 D_K。为了让第(3)步与第(2)步结合，我们再设另一个数列 E_k，使它和 D_k 一一对应，即 D_k 与 E_k 分别代表同一选手在第二第三次竞赛的名次。这样，第(3)步又可以简化为在一段数列 E_k 中查找最小值，即找 $\min(E_k)(k \le K)$

能否在最短时间内查找出这个最小值呢？我们需要一种优化的数据结构来记录 D_k 和 E_k。该数据结构能很快处理 D_k 和 E_k，并且需要支持两种操作。

插入：加入一对数(key,value)到数据结构中。

查询：查询 key 小于 X 的最小的 value

显然，这里的 key 代表 Dk，value 代表 Ek。

因为我们需要的操作与检索树擅长的操作很类似，所以对数据结构有一定理解的选手，在这里都能够想到我们可以构造一个二叉检索树。

【改进六的实现：优化数据结构】我们构造检索树的目的是查询某一区间的最小 value，这样，我们可以定义每个节点代表一个区间，叶子部分代表一个 key，我们依次把每两个叶子指向同一个父亲节点。例如 1 和 2 的父亲是区间[1,2]；[1,2]和[3,4]的父亲是[1,4]，如此类推。我们定义每个节点表示的区间是 key 的区间，而节点值则记录该区间中的最小 value。

插入和查询具体做法如下：

插入操作：当我们接到一个请求：把(a,b)加入到检索树中。那么我们只需要依次从 key=a 的叶子检索到根节点，比较 b 和途中每个节点的值大小，如果 b 小于该节点的值，就用 b 更新它。

查询操作：我们要让检索树能够查询 key 小于 a 的最小 value。显然我们查询[1,a]的最小 value 时，由检索树的性质，我们可以把它分成小区间并分别查询

它。例如当 a=14 时，需要访问到区间[1,8]，[9,12]，[13,13]。对于这题，可以有简单的实现方法：先令 min-value 为 0，依次从表示[a,a]的节点检索到根节点，如果该节点有父亲并且是父亲的右儿子，就比较它的左兄弟和 min-value 的值，如果左兄弟节点的值小于 min-value，就用左兄弟节点的值更新它。到根节点的时候即可以返回 min-value。

这种数据结构能使查询和插入操作都能在 O(LogN)的复杂度下完成，

〖最终算法〗这样我们就能很清晰地得出优化的解法：以第一次竞赛的名次从高到低枚举 X，以第二次竞赛名次为 key，第三次竞赛名次为 value。对于每个 X，只要查询区间[1,key]的最小值 min-value，若 min-value<value，则有选手比 X better，即 X 不是 excellent。反之，若 min-value>value，说明 X 是 excellent，并把(key,value)加入检索树。

由于枚举 X 循环需要 O(N)时间，查询和插入都只需要 O(LogN)，总的时间复杂度是 O(NLogN)。对于所有数据都应该在很短时间内出解。

【小结】在这个部分，我们分析了 Team 一题，过程中总结了很多算法，有的是基本算法，有的是改进不完全的算法。不过很多时候我们思考这一题并不需要这么多的改进，例如我们很容易就能跳过改进一而直接得到改进二的结论。

这一题改进三得出来以后，并没有而且不能直接得出改进四，而需要很重要的一步：分析改进三的本质，从而扩展改进三。

笔者之所以选取了 Team 这题作重点分析，不仅是因为 Team 一题需要灵敏的头脑，全面地观察，还因为它需要对数据结构有一定的认识，笔者当初做这道题的时候就在这里栽了跟头。

四、总结

这篇论文给大家解释了笔者关于规模维数增加的一类问题的看法。由于笔者水平有限，有些道理仍然未能参透，文中的算法还能有改进的地方。但本文旨在阐述在解决这类问题的过程中，应该结合算法的本质和问题的特点。

算法改进的思想其实是开阔进取的思想，即不满足于现状，要开发新的算法，新的思路。人类漫长的历史不就是开阔进取的精神在支配着，没有开阔进取，又何来我们今天的发达的生产力？在 OI 中很多问题我们不能够说随便解决了就 OK，我们还要有开阔进取的精神，不断改进算法和数据结构，才能真正意义上解决问题。

算法改进的思想其实也是创新的思想，即“要创造”的思想。很多时候，信息学奥林匹克的问题并不能直接套用经典算法，而需要加入很多创新的思维，和具体的实践。事实上，任何经典的算法都是由创新的思想总结出来的。只要我们能大胆创新，说不定某一天也会有我们创造的经典算法哩。

【感谢】

感谢林盛华老师指导和杨溢同学的热情帮助。

【参考文献】

[1]《金牌之路 高中计算机竞赛辅导》/江文哉主编.—西安：陕西师范大学出版社，2000.6

[2]《奥赛经典 信息学奥林匹克教程 提高篇》/吴耀斌，曹利国，向期中，朱全民编著.—长沙：湖南师范大学出版社，2001.6

[3]《算法艺术与信息学竞赛》/刘汝佳，黄亮著.—北京：清华大学出版社，2003

[4] Balkan Olympiad Informatic 2004 官方网站提供的原题和解题报告

[5]广东省奥林匹克竞赛 2001（GDOI2001）题目

[6]全国奥林匹克竞赛 2001（NOI2001）题目

【附录】

A.旅行(广东省奥林匹克竞赛 2001)

〖问题描述〗

GDOI 队员们到 Z 镇游玩。Z 镇是一个很特别的城镇，它有 m+1 条东西方向和 n+1 条南北方向的道路，划分成 $m\times n$ 个区域。Z 镇的名胜位于这些区域内。从上往下数第 i 行，从左往右数第 j 列的区域记为 $D(i,j)$。GDOI 队员们预先对这 $m\times n$ 个区域打分 $V(i,j)$(分数可正可负)。分数越高表示他们越想到那个地方，越低表示他们越不想去。为了方便集合，队员们只能在某一范围内活动。我们可以用(m_1,n_l)与(m_2,n_2) $(m_1 \le m_2, n_1 \le n_2)$表示这样的一个范围：它是这些区域的集合：$\{D(i,j) \mid m_1 \le i \le m_2, n_1 \le j \le n_2\}$。GDOI 队员们希望他们活动范围内的区域的分值总和最大。

当然，有可能队员们一个地方也不去（例如，所有区域的分值都是负数。当然，如果某范串内的分值总和为零的话，他们也会去玩）。也有可能他们游览整个 Z 镇。你的任务是编写一个程序，求出他们的活动范围$(m_1,n_l),(m_2,n_2)$。

〖输入格式〗

输入数据存放在当前目录下的文本文件"travel.dat"中。数据有 m+1 行。第一行有两个数 m,n(m,n 定义如上)。其中，$(1\le m\le 250,1\le n\le 250)$。接下来的 m 行，每行 n 个整数，第 i 行第 j 个数表示分值 $V(I,j)$ $(-128\le V(i,j)\le 127)$。每两个数之间有一个空格。

〖输出格式〗

答案输出到当前目录下 "travel.out"中，只有一行，分两种情况:

1．队员们在范围$(m_1,n_1),(m_2,n_2)$内活动，输出该范围内的分值。

2．队员们不想去任何地方，只需输出"NO"。

注意：不要有多余空行，行首行尾不要有多余空格。

〖输入输出举例〗

样例一		样例二	
Travel.dat	travel.out	Travel.dat	travel.out
4 5 1 -2 3 -4 5 6 7 8 9 10 -11 12 13 14 -15 16 17 18 19 20	146	2 3 -1 -2 -1 -4 -3 -6	NO

B.炮兵阵地（NOI2001）

司令部的将军们打算在 $N \times M$ 的网格地图上部署他们的炮兵部队。一个 $N \times M$ 的地图由 N 行 M 列组成，地图的每一格可能是山地（用"H" 表示），也可能是平原（用"P"表示），如下图。在每一格平原地形上最多可以布置一支炮兵部队（山地上不能够部署炮兵部队）；一支炮兵部队在地图上的攻击范围如图中黑色区域所示：

P	P	H	P	H	H	P	P
P	H	P	H	P	H	P	P
P	P	P	H	H	H	P	H
H	P	H	P	P	P	P	H
H	P	P	P	P	H	P	H
H	P	P	H	P	H	H	P
H	H	H	P	P	P	P	H

如果在地图中的灰色所标识的平原上部署一支炮兵部队，则图中的黑色的网格表示它能够攻击到的区域：沿横向左右各两格，沿纵向上下各两格。图上其它白色网格均攻击不到。从图上可见炮兵的攻击范围不受地形的影响。

现在，将军们规划如何部署炮兵部队，在防止误伤的前提下（保证任何两支炮兵部队之间不能互相攻击，即任何一支炮兵部队都不在其他支炮兵部队的攻击范围内），在整个地图区域内最多能够摆放多少我军的炮兵部队。

Input

第一行包含两个由空格分割开的正整数，分别表示 N 和 M；

接下来的 N 行，每一行含有连续的 M 个字符('P'或者'H')，中间没有空格。按顺序表示地图中每一行的数据。N <= 100；M <= 10。

Output

仅一行，包含一个整数 K，表示最多能摆放的炮兵部队的数量。

Sample Input

5 4
PHPP
PPHH
PPPP
PHPP
PHHP

Sample Output

6

C.Team Selection (BalkanOI 2004 Day 1 Task 2)

The Interpeninsular Olympiad in Informatics is coming and the leaders of the Balkan Peninsula Team have to choose the best contestants on the Balkans. Fortunately, the leaders could choose the members of the team among N very good contestants, numbered from 1 to N ($3 \leqslant N \leqslant 500000$). In order to select the best contestants the leaders organized three competitions. Each of the N contestants took part in all three competitions and there were no two contestants with equal results on any of the competitions. We say that contestant A is better than another contestant B when A is ranked before B in all of the competitions. A contestant A is said to be excellent if no other contestant is better than A. The leaders of the Balkan Peninsula Team would like to know the number of excellent contestants.

Write a program named TEAM, which for given N and the three competitions results, computes the number of excellent contestants.

The input data are given on the standard input as four lines. The first line contains the number N. The next three lines show the rankings for the three competitions. Each of these lines contains the identification numbers of the contestants, separated by single spaces, in the order of their ranking from first to last place.

The standard output should contain one line with a single number written on it: the number of the excellent.

EXAMPLE 1

Input

3
2 3 1
3 1 2
1 2 3

Output

3

Note: No contestant is better than any other contestant, hence all three are excellent.

EXAMPLE 2

Input

10
2 5 3 8 10 7 1 6 9 4
1 2 3 4 5 6 7 8 9 10
3 8 7 10 5 4 1 2 6 9

Output

4

Note: The excellent contestants are those numbered with 1, 2, 3 and 5.