浅谈图论模型的建立与应用

广东省中山市第一中学　黄源河

【关键字】 图论模型、建立、转化

【摘要】

在近几年的信息学竞赛中，图论题目层出不穷。图论作为一个新生的数学分支，相比其他数学分支来说，具有许多自有的特性。利用图论解题，通常具有高效、简洁的便利。有了这门工具，并不意味就能很好地解决问题，还在于我们能否熟练地识别与建立一系列的图论模型。本文通过一些实例，简单地介绍一下图论建模的方法。

【正文】

引言

应用数学知识解题时，首先要通过对实际问题的分析，研究组建用以描述这个问题的数学模型。使用数学的理论和方法对模型进行分析从而得到结果，再返回去解决现实的实际问题。图论模型是一类特殊的数学模型，建立图论模型，就是要从问题的原型中，抽取对我们有用的信息和要素，把问题抽象为点、边、权的关系。

经过图论建模之后，杂乱无章的信息变得有规可寻，要素的内在联系体现在了点、边、权的关系。有不少经典的图论模型可以直接用特定的算法解决，一些复杂的问题，只要能认清问题的本质，把握问题的关键，建立合适的图论模型，往往能转化为我们熟悉的经典问题。

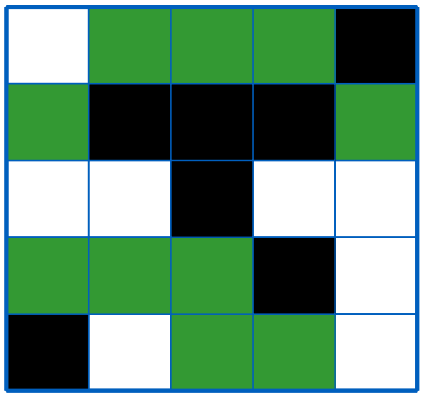
本文要写的，正是我在图论建模方面的一点心得与认识。

例题分析

〖例题 1〗Place the Robots (ZOJ)

［问题大意］

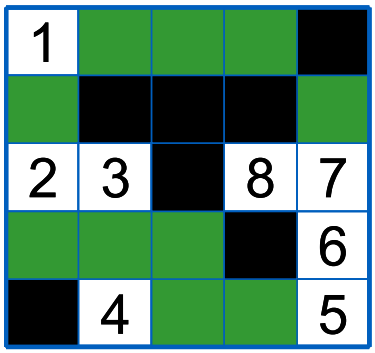
有一个 N*M(N,M<=50) 的棋盘，棋盘的每一格是三种类型之一：空地、草地、墙。机器人只能放在空地上。在同一行或同一列的两个机器人，若它们之间没有墙，则它们可以互相攻击。问给定的棋盘，最多可以放置多少个机器人，使它们不能互相攻击。

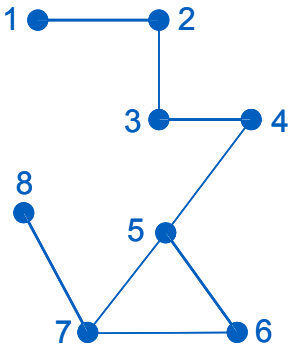


［分析］

在问题的原型中，草地，墙这些信息不是我们所关心的，我们关心的只是空地和空地之间的联系。因此，我们很自然想到了下面这种简单的模型：

以空地为顶点，有冲突的空地间连边，我们可以得到下面的这个图：

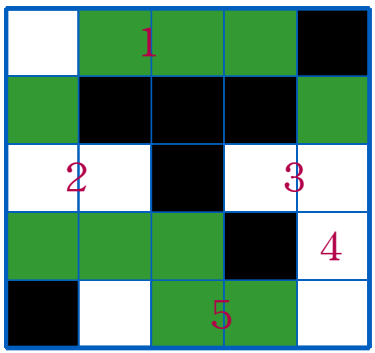


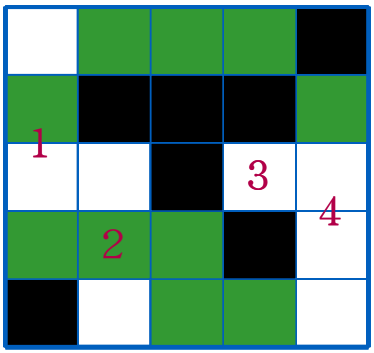


那么，问题转化为求图的最大独立集问题。众所周知，这是 NP-完全问题。

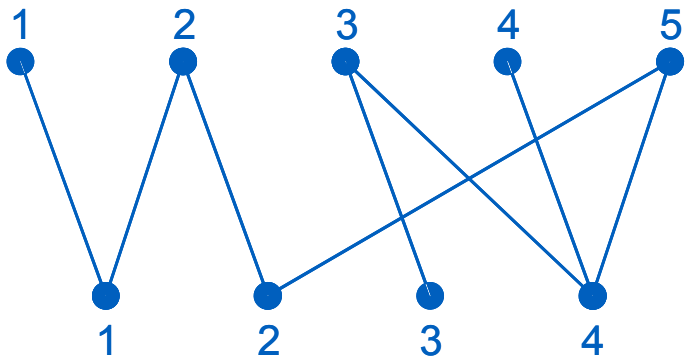
看来，建立这样的模型，没有给问题的求解带来任何便利，我们必须建立一个行之有效的新模型。

我们将每一行，每一列被墙隔开，且包含空地的区域称作“块”。显然，在每个块之中，最多只能放一个机器人。我们把这些块编上号，如下图所示：





把横向块作为 X 部的顶点，竖向块作为 Y 部的顶点，如果两个块之间有公共的空地，就在它们之间连边。这样，我们得到了下面的二部图：



由于每条边表示一个空地，有冲突的空地之间必有公共顶点，所以问题转化为二部图的最大匹配问题。这是图论中的经典问题，可以用匈牙利算法解决。

[小结]

比较上面的两个模型，第一个过于简单，没有认清问题的本质；第二个则充分抓住了问题的内在联系，巧妙地建立了二部图模型。为什么会产生这样截然不同的结果呢？其一是由于对问题分析的角度不同，第一种模型以空地为点，第二种模型以空地为边；其二是由于第一种模型对原型中信息的选取不足，所建立的模型没有保留原型中重要的性质，而第二种模型则保留了原型中“棋盘”这个重要的性质。由此可见，对信息的选取，是图论建模中至关重要的一步。

〖例题 2〗出纳员的雇佣（ACM Tehran 2000）

[问题描述]

Tehran 的一家每天 24 小时营业的超市，需要一批出纳员来满足它的需要。超市经理雇佣你来帮他解决他的问题——超市在每天的不同时段需要不同数目的出纳员（例如：午夜时只需一小批，而下午则需要很多）来为顾客提供优质服务。他希望雇佣最少数目的出纳员。

经理已经提供你一天的每一小时需要出纳员的最少数量——$R_0, R_1, ..., R_{23}$。R_0

表示从午夜到上午 1：00 需要出纳员的最少数目，R_1 表示上午 1：00 到 2：00 之间需要的，等等。每一天，这些数据都是相同的。有 N 人申请这项工作，每个申请者 I 在没 24 小时中，从一个特定的时刻开始连续工作恰好 8 小时，定义 t_I（$0 \le t_I \le 23$）为上面提到的开始时刻。也就是说，如果第 I 个申请者被录取，他（她）将从 t_I 时刻开始连续工作 8 小时。

你将编写一个程序，输入 R_I（I = 0..23）和 t_I（I = 1..N），它们都是非负整数，计算为满足上述限制需要雇佣的最少出纳员数目。在每一时刻可以有比对应的 R_I 更多的出纳员在工作。

输入

输入文件的第一行为测试点个数（<= 20）。每组测试数据的第一行为 24 个整数表示 R_0，R_1，...，R_{23}（$R_I \le 1000$）。接下来一行是 N，表示申请者数目（$0 \le N \le 1000$），接下来每行包含一个整数 t_I（$0 \le t_I \le 23$）。两组测试数据之间没有空行。

输出

对于每个测试点，输出只有一行，包含一个整数，表示需要出纳员的最少数目。如果无解，你应当输出“No Solution”。

[分析]

初看本题，很容易使人往贪心、动态规划或网络流这些方面思考，但这些算法对于本题都无能为力。

由于本题的约束条件很多，为了理清思路，我们先把题目中的约束条件用数学语言表达出来。设 S[i]表示 0～i 时刻雇佣出纳员的总数，Wi 表示在时刻 i 开始工作的申请者的人数。那么我们可以将题目中的约束条件转化为下面的不等式组：

$$\begin{cases} 0 \le S[i] - S[i-1] \le W_i & 0 \le i \le 23 \\ S[i] - S[i-8] \le R_i & 8 \le i \le 23 \\ S[23] + S[i] - S[i+16] \le R_i & 0 \le i \le 7 \end{cases}$$

这样的不等式组，不禁使我们想到了差分约束系统。对于每条不等式 S[i]-S[j]≤K，从顶点 j 向顶点 i 引一条权值为 K 的有向边。我们要求的 S[23]的最小值，只要求顶点 0 到顶点 23 的最短路。但是注意上面第三条不等式：它包含三个未知数，无法在图中表示为边的关系。

思考到这一步，似乎陷入了僵局。难道本题不能用差分约束系统解决吗？不，我们还需要一些转化。退一步海阔天空，如果把 S[23]作为未知数，那是肯定做不下去的。但是如果把 S[23]作为已知数，那么第三条不等式就只有两个未知数 S[i],S[i+16]，我们从顶点 i+16 向顶点 i(0≤i≤7)引一条权值为 R_i-S[23]的边。那么，该不等式组可以完全转化为一个有向图，未知数 S[i]的解，就是图中顶点 0 到顶点 i 的最短路。而当图中存在负权回路时，不等式组无解。

上面的解法是把 S[23]当成了已知数，而实际上 S[23]不但是未知的，而且正是我们要求的。怎么办？我们可以用二分法枚举 S[23]的值，逐步缩小范围，用迭代法判断是否存在负权回路（判定可行性）。如果当 S[23]取到 N 仍不可行，则输出“No Solution”，否则输出 S[23]的最小值。时间复杂度为 $O(24^3*\log_2 N)$。

[小结]

本题用到了差分约束系统的理论，在竞赛中，这样的系统并不多见，但是却可以巧妙的解决一些难题。这类题目的模型都不明显，需要一定的思考和转化。做这类题目，关键是要把题目中的约束条件表示为不等式，再把不等式转化为图的最短路或最长路模型。

〖例题 3〗贪婪之岛（ZOJ）

[问题大意]

有 N（N≤100000）张卡片，每张卡片有三种能力，每种能力的能力值分别为 Ai，Bi，Ci。每张卡片可以使用其中一种能力，且每张卡片只能使用一次。现在需要 A 张卡片使用第一种能力，B 张卡片使用第二种能力，C 张卡片使用第三

种能力（A+B+C≤100）。请计算使用哪些卡片，以及使用卡片的哪项能力，可以使相应的能力值之和最大。

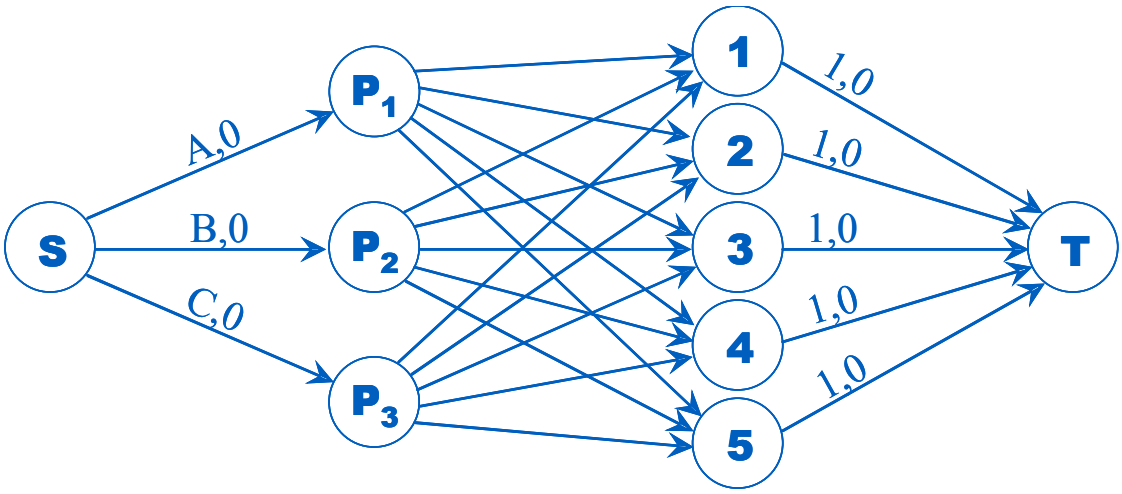
[分析]

最优化问题的解法有很多种，比如动态规划，网络流等，而本题就是一个比较明显的网络流模型。

网络流模型中，权的类型众多，有流量，容量，还可以有费用。在本题中，容量可以作为选取的约束，确保解的合法性；费用则表示选取的价值，确保解的最优性。因此，更确切地说，本题是一个最大费用最大流模型。

认准了问题的模型之后，下一步就是构图了。

- 每张卡片 i 用顶点 i 表示，另外加三个顶点 P_1，P_2，P_3，表示三种能力，还有源点 S，汇点 T。
- 从源点分别向 P_1，P_2，P_3 引一条弧，容量分别为 A，B，C，费用为 0。
- 从 P_1，P_2，P_3 向顶点 i(1≤i≤N) 分别引一条弧，容量为 1，费用分别为 A_i，B_i，C_i。
- 从顶点 i(1≤i≤N) 向汇点引一条弧，容量为 1，费用为 0。



构图之后，求出从 S 到 T 的最大费用最大流，再检查流出 P1，P2，P3 的弧，并输出最优方案。这样的算法是十分粗糙的，时间复杂度为 $O(N^3)$，空间复杂度为 $O(N^2)$，时空均不可行，需要进一步优化。

本题的卡片总数有十万之多，而最终要选取的卡片数不超过 100 张。如果在构图之前，把没有用的卡片先删掉，必将大大提高效率。

什么样的卡片是没有用的呢？

先考虑第一种能力的选取：如果把全部卡片按第一种能力值从大到小排序，显然我们应该尽量从前面选 A 张出来，由于每张卡片只能使用一次，所以有可能会和其他的两种能力发生冲突，而冲突的卡片数最多是 B+C 张，所以实际上对我们有用的卡片只是前面的 A+B+C 张。

同理，对于第二种和第三种能力的选取，也只需保留其能力值最大的前 A+B+C 张卡片。这一步可以在线性时间内解决。

这是一个既简单又有效的方法，经过这一步处理，保留下来的卡片数不会超过 3(A+B+C)张，顶点数大大减少，求解最大费用最大流的时间复杂度降为 $O((A+B+C)^3)$。至此，算法已经优化到了一个可以接受的地步，时间复杂度仅为 $O(N+(A+B+C)^3)$。

如果还要进一步提高效率，可以用更有效的算法删掉多余的顶点。不过这样做意义不大，而且也不是本文讨论的要点。

另外，本题还可以转化为二部图模型，用最佳匹配算法求解。这一步留给读者自己思考。

[小结]

本题建立的是网络流模型。这类模型的算法系数大，编程复杂度也大，在竞赛中往往作为走投无路时的“候补算法”。但是，网络流模型的适用性广，一些较复杂，或者约束较多的问题，网络流模型可以很好地解决，而基于网络流模型的问题又比较明显，这使得网络流模型有着广泛的应用。

【总结】

问题是千变万化的，如何建立问题的图论模型并没有通用的准则。前面的几个例子都比较简单，在更复杂的问题中，有时我们会感到难以建立适当的模型，这时，我们需要在不改变问题原型本身的性质的前提下，对原型进行抽象，简化，在此基础上建立合适，有效的模型。有时，我们建立了问题的一个模型之后，可能会感到难以求解，这时，我们可能需要对模型进行修改，转化，或者对原型进行更深入的分析，抽取其中较关键的要素，建立一个易于求解的模型。这些都需要我们有丰富的经验，灵活的思维以及良好的创造力。

【参考文献】

1．吴文虎、王建德，实用算法的分析与程序设计，电子工业出版社，1998
2．吴文虎、王建德，青少年国际和全国信息学（计算机）奥林匹克竞赛指导——图论的算法与程序设计，清华大学出版社，1997
3．IOI2003 国家集训队第三阶段讨论稿

【附录】

例题 1 原题　题目来源：ZOJ Monthly, October 2003 Contest

Place the Robots

Robert is a famous engineer. One day he was given a task by his boss. The background of the task was the following:

Given a map consisting of square blocks. There were three kinds of blocks: Wall, Grass, and Empty. His boss wanted to place as many robots as possible in the map. Each robot held a laser weapon which could shoot to four directions (north, east, south, west) simultaneously. A robot had to stay at the block where it was initially placed all the time and to keep firing all the time. The laser beams certainly could pass the grid of Grass, but could not pass the grid of Wall. A robot could only be placed in an Empty block. Surely the boss would not want to see one robot hurting another. In other words, two robots must not be placed in one line (horizontally or vertically) unless there is a Wall between them.

Now that you are such a smart programmer and one of Robert's best friends, He is asking you to help him solving this problem. That is, given the description of a map, compute the maximum number of robots that can be placed in the map.

Input

The first line contains an integer T (<= 11) which is the number of test cases.

For each test case, the first line contains two integers m and n (1<= m, n <=50) which are the row and column sizes of the map. Then m lines follow, each contains n characters of '#', '*', or 'o' which represent Wall, Grass, and Empty, respectively.

Output

For each test case, first output the case number in one line, in the format: "Case :id" where id is the test case number, counting from 1. In the second line just output the maximum number of robots that can be placed in that map.

Sample Input

```
2
4 4
o***
*###
oo#o
***o
4 4
#ooo
o#oo
oo#o
***#
```

Sample Output

```
Case :1
3
Case :2
5
```

例题 3 原题（有改动）　题目来源：ZOJ Sunny Cup 2003 Online Contest

Greedy Island

Gon and Killua are engaged in a game of Greedy Island. The goal of the game is to collect 100 spell cards distributed all over the game. A spell card has three properties: Attack, Defense and Special. The numeric value of each property is between 0 and 100. Each card can be used only ONCE. All the spell cards must be stored in the Book - the initial item of the game. The Book can store at most 50 spell cards, so Gon and Killua can have at most 100 spell cards in all. Now Gon and Killua have n spell cards, and they want to use A cards for Attack, B cards for Defense and C cards for Special uses (A + B + C <= 100). If n > A + B + C, they have to discard some cards.

They would like to know the maximum sum of the Attack value in Attack Group, Defense value in Defense Group and Special value in Special Group. If there are multiple solutions, choose the solution with the maximum sum of ALL the properties of all the cards.

Input

The first line contains an integer T (1 <= T <= 10), the number of test cases.

For each test case, the first line contains a single integer n (n <= 100,100); the next line contains three integers A, B and C (A, B, C >= 0, A + B + C <= n); the following n lines contain the Attack value, Defense value and Special value of the n spell cards.

Output

For each test case, print the maximum sum of Attack value in Attack Group, Defense value in Defense Group and Special value in Special Group, and maximum sum of ALL the properties of all the cards in one line, separated with one space.

Sample Input

```
2
3
1 1 1
100 0 0
0 100 0
0 0 100
4
1 1 1
12 32 44
33 48 37
37 38 33
46 79 78
```

Sample Output

```
300 300
163 429
```