拯救 Protoss 的故乡

2009-2010 信息学国家集训队论文

潘宇超

浙江省绍兴市第一中学

2010 年 1 月

摘要

本文主要从一道原创题出发，讲述了这道题的若干种解法。从最大流到动态规划，再从动态规划到最小费用流，并给出了最小费用流解法正确性的证明。接着充分利用题中的条件，对最小费用流进行了优化，得到一个比较优秀的解法。另外，本文还介绍了如何设计这 10 个测试点的，对选手得分的分布进行了估计。最后进行了总结。

关键字

星灵族　星灵英雄　Zeratul　Aiur　行星　网络流　最大流　最小费用流　动态规划　线段树　动态树

目录

1. 问题描述

1.1. 题目描述

在星历 2012 年，星灵英雄 Zeratul 预测到他所在的 Aiur 行星在 M 天后会发生持续性暴雨灾害，尤其是他们的首都。而 Zeratul 作为星灵族的英雄，当然是要尽自己最大的努力帮助星灵族渡过这场自然灾害。

要渡过这场自然灾害，Zeratul 自然要安排很多很多事情，其中一件就是将雨水疏导到大海里去。星灵族在重建家园的时候建造了 N 条河流，这些河流连接了共 N+1 个城市，当然其中包括了星灵首都。城市的编号为 0…N，星灵首都的编号为 0。

当然水流是有方向的，除了星灵首都以外，其余的城市都有且仅有一条河流流入。如果一个城市没有流出去的河流，那么这个城市就是一个沿海城市，可以认为流入这个城市的河流是直接流入大海的。

聪明的星灵族在建造河流的时候是不会让其出现环状结构的，也就是说所有的河流都是能够流入大海的。

每一条河流都是有一个最大流量的，一旦超过这个最大流量，就会发生洪水灾害。因此从星灵首都流入大海的总水流量是有一个最大值的。

例如有 3 条河流：第一条是从城市 0 流向城市 1，最大流量为 4；第二条是从城市 1 流向城市 2，最大流量为 2；第三条是从城市 1 流向城市 3，最大流量为 3。此时从星灵首都(城市 0)流入大海的总水流量最大为 4，方案有两种：

A. 第一条河流的流量为 4，第二条河流的流量为 2，第三条河流的流量为 2。

B. 第一条河流的流量为 4，第二条河流的流量为 1，第三条河流的流量为 3。

由于离暴雨到来还有时间，因此 Zeratul 可以扩大某些河流的最大流量来使得从星灵首都流入大海的总水流量增大。比如将上面这个例子的第一条

河流的最大流量增大 1，那么从星灵首都流入大海的总流水量也可以增大 1，变成了 5。

当然将河流的最大流量扩大是需要时间的，将一条河流的最大流量扩大 1 所需要的时间为 1 天。离暴雨到来还有 *M* 天，因此 Zeratul 也有 *M* 天的时间来扩大河流的最大流量。

不过由于河流周围的地形限制，每条河流并不是能够无限扩大的，因此每条河流的可以扩大到的最大流量也是有限制的。

而此时 Zeratul 正忙着安排别的工作，因此他将这个艰巨的任务交给了你。你需要安排一种方案，使得从星灵首都流入大海的总水流量最大。不过现在你只需要告诉 Zeratul 这个最大值即可。

1.2. 输入格式

输入文件的第一行包含一个正整数 *N* 和一个非负整数 *M*。其中 *N* 表示河流的个数，*M* 表示离暴雨到来还剩下的天数。

接下来 *N* 行，每行四个正整数 *U*、*V*、*A*、*B*。其中 *U* 和 *V* 为该河流所连接的两个城市，且河流的水流方向为从城市 *U* 流向城市 *V*，*A* 和 *B* 表示该河流当前的最大流量和可以扩大到的最大流量的上限。

输入数据保证合法。

1.3. 输出格式

输出文件的仅包含一个整数，表示从星灵首都流入大海的最大总水流量。

1.4. 输入样例

```
5 7
0 1 4 8
0 4 1 6
1 2 2 10
1 3 3 5
4 5 6 6
```

1.5. 输出样例

11

1.6. 样例说明

将第一条河流的最大流量扩大 1，变为 5。

将第二条河流的最大流量扩大 5，变为 6。

第三条河流的最大流量不扩大，仍然为 2。

第四条河流的最大流量不扩大，仍然为 3。

第五条河流的最大流量不扩大，仍然为 6。

此时从星灵首都流入大海的总水流量为 6+3+2=11。

1.7. 数据规模

总共 10 个测试点，每个测试点 10 分，数据范围满足：

数据编号	*N*	*M*	数据编号	*N*	*M*
1	≤100	0	6	≤1000	≤100
2	≤100	1	7	≤1000	≤1000
3	≤1000	1	8	≤10000	≤10000
4	≤100	≤100	9	≤10000	≤1000000
5	≤1000	≤100	10	≤10000	≤1000000

所有测试点均满足 $1 \le A \le B \le 100000$。

1.8. 注意事项

提交源程序文件名：protoss.pas/c/cpp

输入文件名：protoss.in

输出文件名：protoss.out

时间限制：1 秒

空间限制：N/A

2. 出题背景

此题的灵感出题的灵感来源于平时的做题。我发现对于一些需要用费用流算法解决的题目，基本上都是最小费用最大流的，也就是求最大流的情况下的最小费用。

然后我就思考是否可以求在费用不超过一个值的情况下的最大流，发现是可以用最小费用流算法解决的。

此题最初的版本是图，为了使得题目更具新颖性和优美性，于是我就将图改成了树，并且又想到了一个动态规划的算法，以及对最小费用流算法的一个优化。

3. 问题分析

3.1. 模型抽象

本题可以抽象为以下模型：

给一个树形网络，根节点 0 是源点，所有叶子节点都是汇点，每条边有容量，每次操作可以将一条边的容量扩大 1，但不能超过这条边的最大容量。这样的操作最多可以进行 M 次。求操作完以后从源点到所有汇点的最大流。

3.2. 算法一

对于测试点 1，直接做一遍最大流即可，由于图的特殊性——是一颗树，因此不会存在退流情况，而且每次增广必定会使得一条边满流，所以复杂度为 $O(N^2)$，可以通过。

对于测试点 2，需要枚举将哪条边的容量扩大 1，然后每次做一遍最大流，取其中的最大值即可，因此复杂度为 $O(N^3)$。

对于测试点 3，如果做法同上的话是会超时的，因此需要进行优化。我们可以发现，如果最大流时一条边(u,v)没有满流，那么就是说从节点 v 出发到其子树中叶子节点的最大流是小于这条边的容量的；否则的话从节点 v 出发到其子树中叶子节点的最大流就是大于等于这条边的容量的。然后就可以用动态规划进行解决。用 $F(i)$来表示从节点 i 出发到其子树中叶子节点的最大流，有以下转移方程：

- 如果节点 i 是叶子节点，那么有：

$$F(i)=\infty$$

- 如果节点 i 不是叶子节点，设 $S(i)$为节点 i 的子节点的集合，$C(i,j)$表示边(i,j)的容量，那么有：

$$F(i)=\sum_{j\in S(i)}\min\{C(i,j),F(j)\}$$

那么 $F(0)$就是在该网络下的最大流。由于这个动态规划的复杂度为 $O(N)$，因此总的时间复杂度就降为 $O(N^2)$。

所以算法一的时间复杂度为 $O(N^{M+1})$，适用范围为 $M \le 1$ 的情况，期望得分为 30 分。

3.3. 算法二

从算法一求最大流的做法得到启发：可以用动态规划来解决。

用 $F(u,i)$表示在以 u 为根的子树中已经操作了 i 次的情况下，从节点 u 出发到其子树中叶子节点的最大流，那么就有以下转移方程：

- 如果节点 u 是叶子节点，那么有：

$$F(u,i)=\infty$$

- 如果节点 u 不是叶子节点，设 $A(u,v)$为边(u,v)的初始容量，$B(u,v)$为边(u,v)的最大容量，然后对于当前节点 u 的子节点 v，转移需要枚举边(u,v)操作了 j 次，以节点 v 为根的子树中总共操作了 k 次：

$$F(u,i)=\max\{F'(u,i-j-k)+\min\{A(u,v)+j,B(u,v),F(v,k)\}\}$$

其中 $F'(i,j)$ 表示在转移当前子节点 v 之前的值。

该动态规划转移复杂度为 $O(M^2)$，因此总的时间复杂度为 $O(NM^3)$，期望得分为 40 分。

接下来我们来优化这个动态规划。想要在状态上进行优化是非常困难的，因此只能在转移时进行优化了。

经观察可以发现 $\min\{A(u,v)+j,B(u,v),F(v,k)\}$ 是和 i 无关的一个值，如果我们用 $G(v,j+k)$来记录 $\min\{A(u,v)+j,B(u,v),F(v,k)\}$ 的最大值，那么转移就只需要枚举 $j+k$ 的和就可以了，因此转移方程变为：

- $G(v,i)=\max\{\min\{A(u,v)+j,B(u,v),F(v,i-j)\}\}$
- $F(u,i)=\max\{F'(u,i-j)+G(v,j)\}$

其中 $F'(i,j)$ 表示在转移之前的状态值。

将动态规划的转移复杂度优化为 $O(M)$，总复杂度优化到了 $O(NM^2)$，因此算法二的期望得分为 60~70 分。

3.4. 算法三

由于题目中要求的是从源点到所有汇点的最大流，不妨再从网络流的角度来思考。

我们建立这样的一个网络：对于一条边(U,V,A,B)，添加两条从节点 U 到节点 V 的边。第一条边的容量为 A，费用为 0；第二条边的容量为 B-A，费用为 1。然后再添加一个汇点，所有的叶子节点添加一条到这个汇点的边，容量为∞，费用为 0。那么题目可以转化为求总费用不超过 M 的情况下的从源点到汇点的最大流。

设 $Best(x)$为流量为 x 时的最小费用，那么显然有以下不等式成立：

$$Best(x) \le Best(x+1)$$

那么题目中就是要求最大的 x，满足 $Best(x) \le M$ 。

解决此问题可以用最小费用增广路算法(Successive Shortest Path Algorithm，简称 SSP)，也就是说每次找一条费用最小的增广路进行增广。

首先引入一个定理：

> (Klein [1967]) *Let* (G,u,b,c) *be an instance of the* ***Minimum Cost Flow Problem****. A b-flow f is of minimum cost if and only if there is no f-augmenting cycle with negative total weight.*

就是说一个流 f 是最小费用流，其充分必要条件为图中不存在费用和为负数的圈。

设用最小费用增广路算法在流量为 x 时的费用为 $Cost(x)$。接下来用数学归纳法来证明对任意非负整数 x，都有 $Cost(x)=Best(x)$。

证明：

当 $x=0$ 时，$Cost(0)=Best(0)=0$，命题成立。

假设当 $x=k$ 时，命题成立，即 $Cost(k)=Best(k)$。

那么当 $x=k+1$ 时，假设 $Cost(x)>Best(x)$，由以上定理可知，图中存在一个费用和为负数的圈。由于 $Cost(x-1)=Best(x-1)$，那么产生的这个费用和为负数的圈其中的一部分必定是在最后找到的那条费用最小的增广路上的。设这一部分是增广路上从节点 u 到节点 v 的，且这一段的费用和为 a。由于增广

路是从节点 u 到节点 v 的，那么在圈上的方向就是从节点 v 到节点 u 的，因此在圈上这部分的费用和就是-a。设圈上从节点 u 到节点 v 的这剩下一部分的和为 b，那么就有-a+b<0，移项得到 b<a。那么同样是从节点 u 到节点 v 的路径，却选择了费用和较大的那条路径，这与“费用最小的增广路”矛盾。

由上所述，假设不成立，因此 $Cost(x) \le Best(x)$。又由于 $Best(x)$是最优值，所以 $Best(x) \le Cost(x)$。

那么最终就得到 $Best(x)=Cost(x)$。

证毕。

算法实现如下：

```
Ans ← 0
while (find path between S and T) do
  P ← a shortest path between S and T
  D ← sum of cost in P
  C ← the flow can expand along P
  if D*C>M then
    Ans ← Ans + M div D
    break
  else
    Ans ← Ans + C
    M ← M – D*C
  end if
end while
return Ans
```

由于图是一棵树，根到任意一个叶子节点存在且仅存在一条路径，因此每次增广都能至少增加一条满流边，所以增广次数仅为 $O(N)$。又由于每次增广的复杂度为 $O(N)$，因此该算法的总时间复杂度为 $O(N^2)$，期望得分 100 分。

值得一提的是，该算法的应用范围不仅仅是树，在任意图中也是可以的，但应用在图中的话此算法的时间复杂度会更高。

3.5. 算法四

算法四是在算法三基础上，根据树的特殊性进行了优化。

设 $D(v)$为到叶子节点 v 的最小费用，那么每次增广就是找一个 D 值最小的叶子节点来增广。

由于每次增广都会使得至少一条边满流，设这条满流边为是从点 u 到点 v 的。如果这条满流边的费用为 0，那么还有一条从点 u 到点 v 的费用为 1 的边存在，因为是最小费用增广路，所以这条费用为 1 的边肯定是还没流过的，因此需要将以点 v 为根的子树所有叶子节点的 D 值都增加 1。而如果这条满流边的费用为 1，那么节点 u 就无法再到达节点 v 了，因此需要将以点 v 为根的子树所有叶子节点的 D 值都设为∞。

按照深度优先搜索将所有叶子节点按访问顺序记录在一张表中，那么一个节点的子树包含的所有叶子节点在表中就是连续的一段。

下面给出伪代码：

```
DFS(u)
  L[u] ← Tot+1
  For v be every son of u do
    DFS(v)
  if u is a leaf node then
    Tot ← Tot + 1
  R[u] ← Tot
```

然后我们要支持以下操作：

A. 查询一段区间中的最小值。

B. 将一段区间中的值都加上一个值。

C. 查询叶子节点到根的路径中权值最小的边。

D. 修改某条边的权值。

E. 将叶子节点到根的路径中的边权都减少一个值。

对于操作 A、B，用线段树维护即可。

对于操作 C、D、E，可以用动态树实现。

对于每次操作，线段树的复杂度为 O(log N)，而用 Splay 实现的动态树均摊复杂度为 O(log N)。

所以算法四的时间复杂度为 O(N log N)，期望得分为 100 分。

4. 数据设计

4.1. 测试点 1

N=98，*M*=0。

随机。

4.2. 测试点 2

N=99，*M*=1。

满二叉树。

4.3. 测试点 3

N=989，*M*=1。

随机。

4.4. 测试点 4

N=100，*M*=97。

从根连出去 10 条链，每条链长度都为 10。

4.5. 测试点 5

N=1000，*M*=99。

从根连出去 200 条链，每条链长度都为 5。

4.6. 测试点 6

N=1000，*M*=100。

先构造一个长度为 99 的链，其余的节点随机。

4.7. 测试点 7

N=1000，*M*=1000。

满二叉树。

4.8. 测试点 8

N=10000，*M*=9999。

所有节点都与根相连。

4.9. 测试点 9

N=10000，*M*= 991210。

首先构造一条长度为 *N*/2 的主链，然后剩下的 *N*/2 个节点都连到主链的末端的那个节点上。

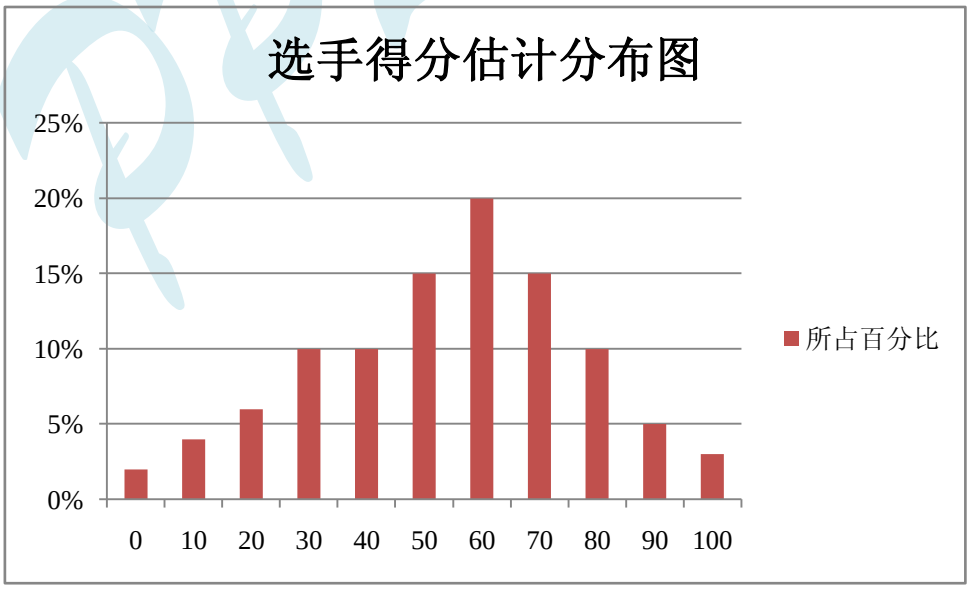
4.10.测试点 10

N=10000，*M*= 994615。

首先构造一条长度为 *N*/2 的主链，然后主链上每个节点都往外连一个节点。

5. 选手得分分布估计

选手得分	所占百分比
0	2%
10	4%
20	6%
30	10%
40	10%
50	15%
60	20%
70	15%
80	10%
90	5%
100	3%



6. 总结

通过本次出题，让我得到了一次宝贵的经验，了解了出题的艰辛，其中的每一步环节都是非常重要的。例如，题目描述要使得模型不那么容易被看出来，解题算法必须正确且严谨，数据设计要使得对不同的算法有区分度，等等。

最优算法(算法四)中由于出现了“动态树”这个东西，相信没有多少人是会的。虽然本题中也可以用树链剖分来做，不过考虑到编程复杂度，也就没出大范围的数据了，因此只需要用到算法三即可，并且这也是希望能有更多人得到高分。

参考文献

1. Ravindra K.Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithm, Applications*. Prentice Hall, United States ed edition. February 1993, pp.294-356.
2. Bernhard Korte, Jens Vygen. *Combinatorial Optimization Theory and Algorithm* (*Third Edition*). Springer. May 2005, pp.157-214.
3. Morton Klein. *A primal method for minimal cost flows with applications to the assignment and transportation problems*. Management Science. 1967.
4. 刘汝佳，黄亮.《算法艺术与信息学竞赛》. 清华大学出版社. 2004 年 1 月，第 315-328 页.
5. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Introduction to Algorithm* (*Second Edition*). The MIT Press. May 2002, pp.643-700.
6. Daniel D. Sleator, Robert Endre Tarjan. *A data structure for dynamic trees*, Journal of Computer and System Sciences. June 1983, pp.362-391.
7. Daniel D. Sleator, Robert Endre Tarjan. *Self-adjusting binary trees*. Proceedings of the fifteenth annual ACM symposium on Theory of computing. Dec 1983, pp.235-245.