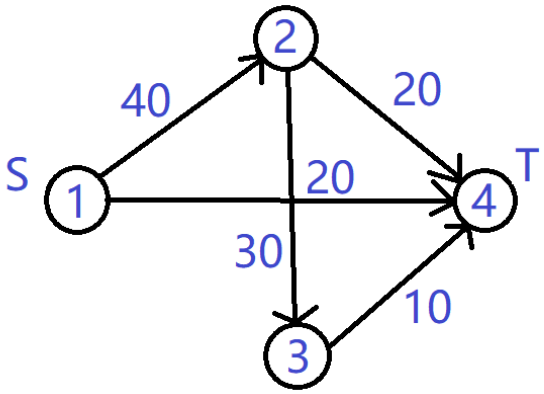
乌克兰大野猪
以梦为马 不负韶华

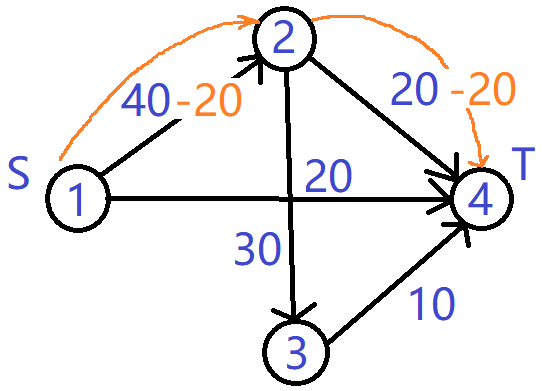
（通俗易懂小白入门）网络流最大流——EK算法

网络流

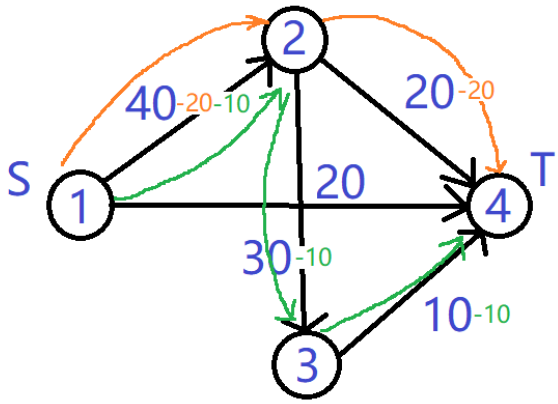
网络流是模仿水流解决生活中类似问题的一种方法策略，来看这么一个问题，有一个自来水厂S，它要向目标T提供水量，从S出发有不确定数量和方向的水管，它可能直接到达T或者经过更多的节点的中转，目前确定的是每条水管中水流的流向是确定的（单向），且每个水管单位时间内都有属于自己的水流量的上限（超过会爆水管），问题是求终点T单位时间内获得的最大水流量是多少？如下图：



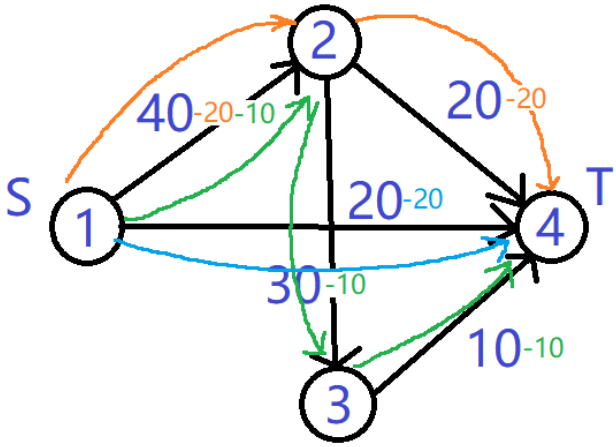
1. 首先，我们用正常的思路去解决这个问题，对于上图的情况而言，我们可以先选择一条水流的路线1->2->4，而且我们得知1->2水管水流量上限为40，而2->4水管水流量上限只有20，则这一条路径通往终点T的水流最大值为20，而这么走的话，1->2的水管中剩余水流量则为40-20=20，2->4水管中的剩余水量为20-20=0，注意：这里水管剩余流量为0后说明这条路径就再也不能接通了，或者说它已经被完全占用了，此时终点T单位时间获得水量0+20=20



2. 然后我们继续选择一条不包含0流量的从S到T的路径，比如1->2->3->4，此时1->2上水流量剩余20，2->3剩余30，3->4剩余10，很显然这条路径上最终能为终点T提供的单位时间的水量由路径上的最小剩余流量10决定（就像是木桶的短板效应一样），这么走的话，1->2水管中剩余的水量为40-20-10=10，2-3水管中剩余的水量为30-10=20，3->4水管中剩余的水量为10-10=0，它也被完全占用了，此时终点T单位时间获得水量为20+10=30



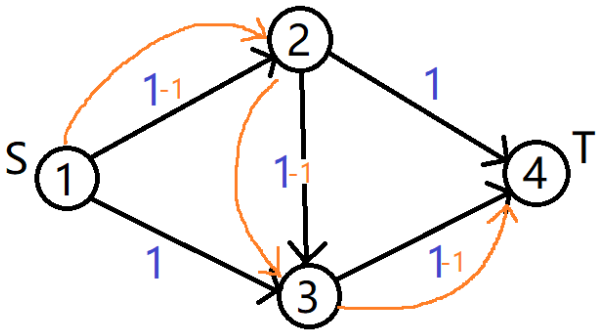
3. 我们继续找还能找到一条从S流向T的路径1->4，水流上限20，因为是直达的，所以路径上最小剩余流量就是它本身，则这么走的话，1->4上的水流量全部耗尽，20-20=0，它也完全被占用了，而终点T再一次获得20水量，30+20=50，至此整张网络流图再也找不到一条能从S到T的不包含0流量的通路，终点T单位时间获得的最大水流也计算出来得到50



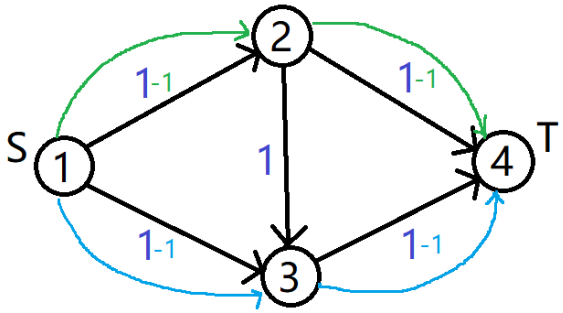
但是，上述的推理只是碰巧数字比较合适让我们轻而易举得到了**50**这个看似

正确的结果，实际上存在着缺陷，假如我第一次找的一条路径并不是此时的最优解，并且这么选使得我下一次选择别的路径的时候有的水管流量已经被占完了，我想反悔怎么办

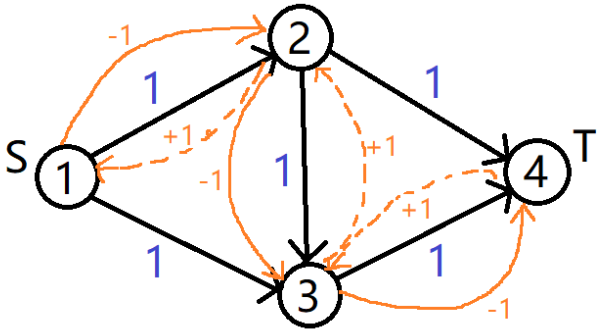
如下图：如果我们第一次选择的是1->2->3->4这条路径（这很符合程序设计的观念，计算机很容易就按照序号去查找），那么1->2水管剩余流量为1-1=0，2->3水管剩余流量为1-1=0，3->4水管的剩余流量为1-1=0，至此我们发现所有的通路都被这三个0流量的水管阻断了，而此时终点T获得的单位水量只有0+1=1



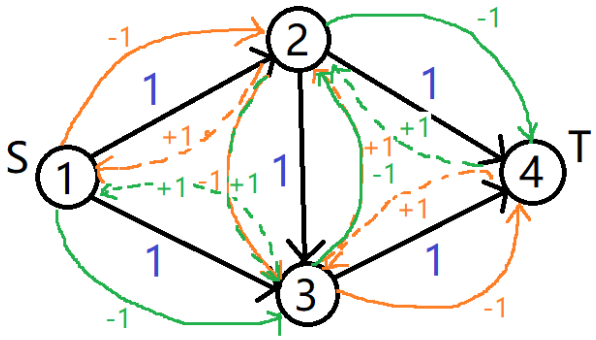
按正常的人为的思路，此时我想要反悔之前选的那条路径，同时选择别的路径，而下图的1->2->4和1->3->4这两条路径的选择才是这张网络流图的最优选择，终点T得到的最大水量为1+1=2，而2->3的这条水管我们放弃不用



上述反悔的过程在我们的思维看来确实可以（假装）没有走过2->3这条路，但是计算机并不能这么去主动理解这种情况，它应该按并不是最优的情况去尝试，当遇到可能更优的情况时更改之前的选择，在我们设计程序的时候，如何能做到这个反悔的过程呢，关键来了：我们对于每次选择的一条道路上的每个水管都要减去路径上的最小残量的同时，为这两个相邻的点添加一条反向弧，反向弧的数值加上最小残量的数值，如上上图那个并不是最优的走法，假设计算机确实第一次就选择了这条路径，那么我们做出如下处理：1->2剩余流量依旧为1-1=0，而同时添加一条2->1的反向弧，流量为0+1=1（反向弧也是一条通路，它的存在就像是为我们提供了一个返回的机会），2->3剩余流量为1-1=0，添加3->2的反向弧，剩余流量为0+1=1，同样3->4的剩余流量为1-1=0，而4->3的反向弧上的剩余流量为0+1=1，所以本次选择之后终点T依旧获得了1的单位流量



接下来我们继续选择一条路径，1->3->2->4（当然也只剩下这条了），此时我们如上述操作，为1->3剩余流量，3->2剩余流量，2->4剩余流量都减去这条路径上的最小残量1，同时为它们两两之间的反向弧增加上这个最小残量1（图可能有点乱，但是原理是不变的，实线方向-，虚线方向+），至此，1->2=0，2->1=1，1->3=0，3->1=1，2->3=1，3->2=0，2->4=0，4->2=1，3->4=0，4->3=1，从S到T已经没有一条额外的路径不包含0流量了，此时T获得最大单位水量2，并且通过这种建立反向弧的方式，我们对于从2到3再从3到2都走了一遍，这不就好像是模拟了一遍反悔的过程吗（具体证明这里不作详述），接下来讲解EK算法



铺垫了这么多终于要讲解EK算法了

别着急在讲解**EK**之前我们需要介绍几个网络流中最重要的概念，我们称起点**S**，也就是水流的出发点为源点，将水流的终点**T**称为汇点，而我们每一次找一条从源点到汇点的不包含**0**流量的路径称为增广路，（增广路顾名思义，如果能找到一条从**S**到**T**的增广路，则到终点的水流量一定还可以增加至少**1**，所以就满足了增广这个要求了），其实求网络流最大流的问题的各个算法都是在模拟一个找寻增广路的过程，如果找不到了就说明此时终点获得的水量将无法变得更大，答案就算出来了

EK算法：从**S**点出发不断找一条到**T**的增广路的过程，我们通过**BFS**向周围搜索与**S**直接相连的剩余流量不为**0**的节点（这个节点一定要是没走过的，因为一条增广路每个点肯定值出现了一次），将他们加入队列，每次从队列中取出一个元素继续向周围查询，直到目标点为**T**点，且这一条道路上不包含流量为**0**的水管，则说明这是一条增广路，为沿途的所有节点两两之间的剩余流量减去该条增广路的最小残量，而同时为它们的反向弧加上最小残量，不断循环直到无法从**S**点到**T**找到一条增广路为止

这里推荐一题网络流最大流的模板题，**POJ1273**，题面讲的是有**n**个水管，有**m**个点，源点为**1**，汇点为**m**，求汇点**T**单位时间的最大水流量，当然这题有个小坑，就是输入会重复，如果**1->2 40**代表从**1**流向**2**有**40**流量，那可能会有多次**1->2 40,1->2 30**之类的，要累加成**1->2 70**

代码：

```
#include<iostream>
#include<stdio.h>
#include<queue>
#include<string.h>
using namespace std;

const int N = 205;
const int INF = 0x3f3f3f3f;
int c[N][N];         //记录i到j的剩余流量
int p[N];             //p[i]记录流向i点的前驱节点
int v[N];             //记录在一条增广路中，流到i点时，此刻增广路上残余量的最小值，直到i == m时就是整条增广路上的残余量最小值
int n, m;

int min(int a, int b){
    return a <= b ? a : b;
}

void EK(){
    //从1出发，不断找可以到达m的增广路
    int ans = 0;
    while(true){
        //EK算法的核心是通过bfs不断查找增广路，同时建立反向弧
        //每次循环都要对v数组和p数组进行清空，因为是意图查找一条新的增广路了
        memset(p, 0, sizeof(p));
        memset(v, 0, sizeof(v));
        queue<int> q;
        q.push(1);
        v[1] = INF;
        //每次只找一条增广路，同时修改c[i][j]的值
        while(!q.empty()){
            int f = q.front();
            q.pop();
            for(int i = 1; i <= m; i++){
                if(v[i] == 0 && c[f][i] > 0){         //v[i]原本是记录增广路实时的残量最小值，v[i]==0代表这个点还没有走过，且从p到i的残量大于0说明通路
                    v[i] = min(v[f], c[f][i]);         //实时更新v[i]的值，v[f]存储1条增广路中i点前所有水管残量的最小值，v[i]为该条增广路到i点为止，路径上的最小残量
                    p[i] = f;                          //p[i]实时保存i点的前驱节点，这样就当i==m时整条增广路就被记录下来
                    q.push(i);                         //将i点入队
                }
            }
        }
        if(v[m] == 0) break;                           //如果v[m]==0则代表找不到增广路了（中途出现
```

```
了c[i][j]==0的情况)
42          ans += v[m];
43          int temp = m;
44          while(p[temp] != 0){                        //类似并查集的查操作，不断查上一个元素且将剩余残
量减去最小残联，反向弧增加最小残量
45              c[p[temp]][temp] -= v[m];
46              c[temp][p[temp]] += v[m];
47              temp = p[temp];
48          }
49      }
50      printf("%d\n", ans);
51  }
52
53  int main(){
54      while(scanf("%d%d", &n, &m) != EOF){
55          memset(c, 0, sizeof(c));
56          for(int i = 1; i <= n; i++){
57              int x, y, z;
58              scanf("%d%d%d", &x, &y, &z);
59              c[x][y] += z;                   //初始时，从x流向y的剩余流量就是输入的值
60          }
61          EK();
62      }
63      return 0;
64  }
```

分类: 网络流

标签: 网络流最大流, EK算法

好文要顶 关注我 收藏该文

乌克兰大野猪
关注 - 1
粉丝 - 3
+加关注

1 推荐　0 反对

« 上一篇： 【NOI 2015】程序自动分析 并查集与离散化处理
» 下一篇： （通俗易懂小白入门）二分图最大匹配——匈牙利算法

posted on 2019-08-07 13:49 乌克兰大野猪 阅读(1055) 评论(1) 编辑 收藏

评论:

#1楼 2019-12-03 03:53 | Kevin_n
第44行错了，应该是“while(temp != 0){”吧 :D

支持(0) 反对(0)

回复 引用

刷新评论 刷新页面 返回顶部

发表评论

编辑　预览

提交评论　退出　订阅评论

[Ctrl+Enter快捷键提交]

相关博文：
· 最大流EK算法
· 网络流--最大流
· 网络流算法
· Dinic算法（研究总结，网络流）
· 最大网络流的——EK算法
» 更多推荐...

昵称: 乌克兰大野猪
园龄: 9个月
粉丝: 3
关注: 1
+加关注

git使用杂谈(2)
PAT甲级(22)
暴力(1)
并查集(1)
动态规划（DP）(4)
二分(1)
二分图(1)
计算机网络(1)
记忆化搜索(1)
矩阵快速幂(3)
模拟(3)
树状数组(1)
数论(2)
贪心(2)
天梯赛题集(8)
网络流(1)
线段树(1)
字符串Hash(1)

随笔档案

2020年2月(12)
2020年1月(2)
2019年11月(11)
2019年10月(23)
2019年8月(9)

最新评论

1. Re:树状数组入门(简单的原理讲解)

感谢楼主！一下就看明白了！

--Wonder007

2. Re:C/C++解题常用STL大礼包 含vector，map，set，queue（含优先队列），stack的常用用法

我记得曾经有个鬼畜里面我见过你的名字[狗头]

--vipwp

3. Re:树状数组入门(简单的原理讲解)

赞一个！楼主写得好通俗易懂，连我这种小白都看懂了

--林学徒

4. Re:树状数组入门(简单的原理讲解)

@xiongyuqing 估计是楼主写错了，不应该那么写。...

--cupid~~

5. Re:树状数组入门(简单的原理讲解)

main函数里输入了a[i]，建树的时候updata函数里a[i]又加上value，那岂不是a[i]+=a[i]?

--xiongyuqing