网络流_13 费用流

费用流

定义:

给定一个流网络，给每条边赋予一个花费，为`w(u,v)`费用/流.定义一个网络的可行流的费用为:
$\sum f(u,v) \cdot w(u,v)$

> 注意，在增广路径中费用应该取反，这样方便退流

算法1 EK算法求最最小费用最大流

板子

> 此算法无法处理具有负权回路的图，否则需要使用消圈法

思路

将`BFS`找增广路改为`SPFA`找一条最短的增广路即可

Code

```
/*链式前向星存图的好处是可以快速找到反向边 edge^1*/
int n,m,S,T;
int h[N],e[M],w[M],f[M],ne[M],idx; //f[]存储残留网络容量
int q[N],d[N],pre[N],incf[N]; //q[]为bfs用队列,d[]为增广路径的残存容量,pre[N]为记录的反向边
bool st[N];         //incf[u]表示到u点的最大流量

void add(int a,int b,int c,int d){
    e[idx] = b, f[idx] = c, w[idx] = d, ne[idx] = h[a] , h[a] = idx++;
    e[idx] = a, f[idx] = 0, w[idx] = -d, ne[idx] = h[b] , h[b] = idx++;
}
/*把bfs换成spfa就是求最小费用最大流的方法*/
bool spfa(){
    int hh=0,tt = 1;
    memset(d, 0x3f, sizeof d);
    memset(incf, 0, sizeof incf);
    q[0] = S, d[S] = 0, incf[S] = INF;
    while(hh != tt ){  //由于spfa会使得每个点入队多次，因此需要使用循环队列
        int t = q[hh ++];
        if(hh == N) hh = 0;
        st[t] = false;

        for(int i=h[t];~i;i=ne[i]){
            int ver = e[i];
            if(f[i] && d[ver] > d[t] + w[i]){
                d[ver] = d[t] + w[i];
                pre[ver] = i;
                incf[ver] = min(f[i],incf[t]);
                if(!st[ver]){
                    q[tt ++ ] = ver;
                    if(tt == N) tt = 0;
```

```
                    st[ver] = true;
                }
            }
        }
    }
    return incf[T] > 0;
}

void EK(int &flow,int &cost){
    flow = cost  =0;
    while(spfa()){
        int t = incf[T];  //走到终点时的最大流量
        flow += t, cost += t*d[T];
        for(int i=T;i!=S;i=e[pre[i]^1]){
            f[pre[i]] -= t, f[pre[i]^1] += t;
        }
    }
}
```

运输问题

> W公司有m 个仓库和 n 个零售商店。
>
> 第 i 个仓库有 a_i 个单位的货物；第 j 个零售商店需要 b_j 个单位的货物。
>
> 货物供需平衡，即$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n}(b_j)$。
>
> 从第 i 个仓库运送每单位货物到第 j 个零售商店的费用为 $c_{i,j}$。
>
> 对于给定的 m 个仓库和 n 个零售商店间运送货物的费用，计算最优运输方案和最差运输方案。

做法：建立超级源汇点然后（1）从源点向仓库连仓库容量花费0的边（2）从仓库向商店连容量INF花费$c[i][j]$的边（3）从商店向汇点连容量为商店需求，价格为0的边，跑最小费用最大流即可

求最大值：先将所有边还原 `f[i] += f[i^1], f[i^1] = 0`,然后将所有边的花费取反。

注意事项，在双向边中不能进行合并，因为反向边为了退流导致费用为负

二分图最优匹配

> n个人n件物品，每个人对应很多件物品，对应每一件物品都有一个满意度。求最大满意度（保证是一个完美匹配）

- 思路

将所有花费去负后用二分图的建图方式跑最小费用最大流即可

数字梯形问题

2 3
3 4 5
9 10 9 1
1 1 10 1 1
1 1 10 12 1 1

> 给定一个由 n 行数字组成的数字梯形如下图所示。
>
> 梯形的第一行有 m 个数字。
>
> 从梯形的顶部的 m 个数字开始，在每个数字处可以沿左下或右下方向移动，形成一条从梯形的顶至底的路径。
>
> 规则 1：从梯形的顶至底的 m 条路径互不相交。
>
> 规则 2：从梯形的顶至底的 m 条路径仅在数字结点处相交。
>
> 规则 3：从梯形的顶至底的 m 条路径允许在数字结点相交或边相交。

建图方式:

`rule 1`

$对每个点的限制 \rightarrow 拆点, 点之间容量限制为1$

$对每条边的限制 \rightarrow 每条点与点之间的容量限制为1$

`rule 2`

$对每个点的限制 \rightarrow 拆点, 点之间容量限制为INF$

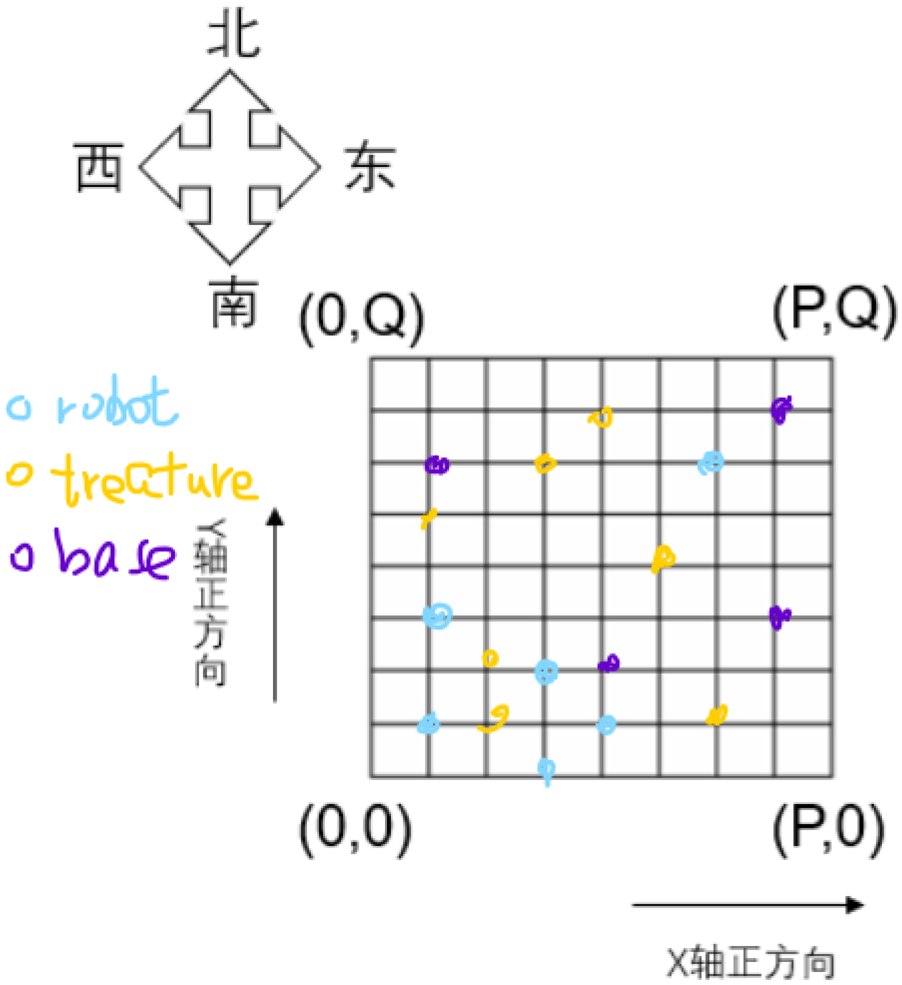
$对每条边的限制 \rightarrow 每条点与点之间的容量限制为1$

`rule 3`

$对每个点的限制 \rightarrow 拆点, 点之间容量限制为INF$

$对每条边的限制 \rightarrow 每条点与点之间的容量限制为INF$

深海机器人问题



> 地图上给出所有机器人的坐标和宝藏的坐标并给出机器人的目的地，限制机器人不能往回走(向下或者向左)，机器人必须到达基地，否则携带的的宝藏不能进行回收，没有价值。每个基地有一定的容量，即限制机器人到达此基地的个数。问机器人的最优移动方案是的尽可能多的机器人到达目的地。

- 建图思路

建立虚拟源点`S`和汇点`T`，从S向所有机器人连容量为机器人个数的边，从所有基地向汇点连容量为基地容量的边。然后根据网格图的建图方式，如果某个地方有宝藏，就建两种边`add(i,j,1,-val)`和`add(i,j,INF,0)`然后最大费用最大流即为所求。

餐巾计划问题

> 一个餐厅计划未来n天的餐巾安排。第i天需要r_i的餐巾。购买一块餐巾需要p元，送到快洗店需要`fc`元，需要等待`fd`天才能洗好。送到慢洗店需要`sc`元，需要等待`sd`天才能洗好。问合理安排n天的计划，最少花费是多少

- 建图思路

理一下关系：

第一类点：每一天需要使用r_i条毛巾，因此从第i天向汇点T连容量为r_i的边。

第二类点：每一天会产出r_i条毛巾，可以分别通过快洗店和慢洗店通过一定的花费转移到第`i+sd`和`i+fd`天，或者直接购买花费p元

因此我们可以将每一天拆出第一类点和第二类点，第一类点向汇点连边，源点向第一类点连边

NOI2008志愿者招募

> 经过估算，这个项目需要 NN天才能完成，其中第 i 天至少需要 Ai 个人。布布通过了解得知，一共有 M 类志愿者可以招募。其中第 ii 类可以从第 Si 天工作到第 Ti 天，招募费用是每人 Ci 元。
>
> 问最小花费

- 错误建图:

源点向每个人连边，每类人向能够服务的天连边，每天向汇点连边。

错误原因：每个人工作不连续而且不好分配费用。

- 正确建图

实际上是一个有下届的最小费用最大流。每一天都有一个下届，因此按照无源汇有下届的建图方式进行建图即可。