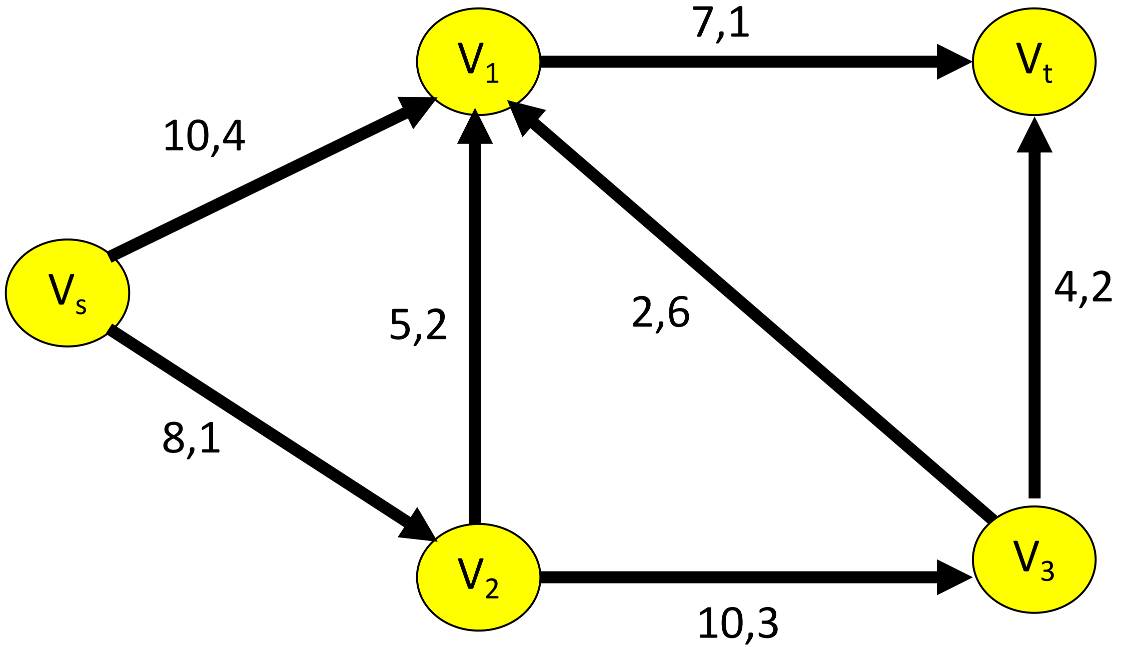
图论算法

费用流

成都市石室中学 梁德伟

费用流模型

(容量C_{ij},单位流量费用B_{ij})



输送费用：B(F)= $\sum_{(V_i,V_j)} B_{ij} * F_{ij}$

费用流模型

– 最小费用增广路算法。
– 定理：如果残量网络中无负圈，则增广最小费用增广路之后，残量网络中也无负圈
– 定理：如果初始网络中费用无负圈，则执行最小费用增广路算法的过程中始终不产生负圈。

将标号法中的“可改进路”变成“最小费用可改进路”

具体的：用最短路算法代替bfs，增广时，只考察dist(u) =dist(v) + costuv的边

记dist(x)表示点x在残量网络中到汇点t的距离

ZKW算法
Dinic式标号修改

```
int spfa(int s,int t){
    memset(vis,0,sizeof(vis));
    memset(dis,0x3f,sizeof(dis));
    dis[t]=0;vis[t]=1;
    deque<int>q;
    q.push_back(t);
    while(!q.empty()){
        int now=q.front();q.pop_front();vis[now]=0;
        for(int i=first[now];i;i=e[i].nxt){
            if(e[i^1].c && dis[now]-e[i].f < dis[e[i].v] ){
                dis[e[i].v]=dis[now]-e[i].f;
                if(!vis[e[i].v]){
                    vis[e[i].v]=1;
                    if(!q.empty()&&dis[e[i].v]<dis[q.front()])q.push_front(e[i].v);else q.push_back(e[i].v
                }
            }
        }
    }
    return dis[s]<0x3f3f3f3f;
}
int cflow(){
    int ret=0;
    while(spfa(s,t)){
        vis[t]=1;
        while(vis[t]){
            memset(vis,0,sizeof(vis));
            ret+=dfs(s,inf);
        }
    }
    return ret;
}
```

```
int dfs(int x,int f){
    if(x==t){
        vis[t]=1;return f;
    }
    int used=0,w;vis[x]=1;
    for(int i=first[x];i;i=e[i].nxt){
        int v=e[i].v,c=e[i].c,ff=e[i].f;
        if(!vis[v] && c && dis[x]-ff==dis[v]){
            w=dfs(v,min(c,f-used));
            if(w)ans+= w*ff, e[i].c-=w;e[i^1].c+=w;used+=w;
            if(used==f)break;
        }
    }
    return used;
}
```

小结

在找增广路时优先找费用最小的流，便是最小费用最大流。优先找最大费用的流，便是最大费用最大流

2635 最小费用流模板

- 套模板

费用流建模

- 最大流代表一种可行方案
- 将费用附在一条最大流上就代表一种可行方案的最小/大费用

看一个常规最优化问题1851 传纸条

1851 传纸条

- 在一个矩形上求两条不相交的价值最大的路径

1851 传纸条

建模过程：

1、找出方案可行：考虑最大流建图方法

首先因为要找不想交的两条路径，我们可以限制每个点只被选择一次，对于点的限制一般是拆点的套路。

将一个点i拆分为2个点，i和i’并且连边权为1,这样就可以限制这个点只被选择一次，不过源点和汇点（左上角、右下角）的两个点可以选择两次所以边权为2

然后i’和下方的点j连边，i’和右边的点i连边，边权都为1，表示可以向下走，和向右走。边权为1表示只能走一次

1851 传纸条

建模过程：

2、考虑费用赋权

对于原图，只有点权。所以将i和i’之间的边赋权值为点权，其他的费用为0

最后跑一次费用流即可

建模代码参考

```
inline int idx (int x,int y){
    return (x-1)*m+y;
}
int dx[]={0,1 };
int dy[]={1,0 };
int main(){
    n=in;m=in;int tmp=0;
    for(int i=1;i<=n;i++)
    for(int j=1;j<=m;j++){
        int x=in;
        int id=idx(i,j);
        if(id==1 || id==m*n)ins(id,id+n*m,2,x),tmp+=x;
        else ins(id,id+n*m,1,x);
        for(int k=0;k<2;k++){
            int tx=i+dx[k],ty=j+dy[k];
            if(tx==n+1||ty==m+1)continue;
            int id1=idx(tx,ty);
            ins(id+n*m,id1,1,0);
        }
    }
    int ans=0;S=1,T=2*n*m;
    D::flow(S,T,ans);
    cout<<ans-tmp;
    return 0;
}
```

Bzoj1283序列-woj4267

- 给出一个长度为 的正整数序列Ci，求一个子序列，使得原序列中任意长度为m 的子串中被选出的元素不超过K(K,M<=100) 个，并且选出的元素之和最大。

分析

利用最大流代表一个可行方案，考虑如下建图：

建立源、汇S,T。对序列每个元素建一个点ai

S向a1连一条容量为k的边，费用为0

Ai向ai+1连一条容量为k，费用为0（表示不选ai到a_(i+1)）

An向T连一条容量为k，费用为0

Ai(1<=i<=n-m)向a_(i+m)连一条容量为1，费用为ai。（表示选择了ai）

Ai (n-m<i<=n)向T连一条容量为1，费用为ai（表示选择了ai）

可以证明，转换后的可行方案是原问题的可行方案

2626「网络流 24 题」餐巾计划

- 一个餐厅在相继的 n 天里，每天需用的餐巾数不尽相同。假设第 i天需要 ri 块餐巾。餐厅可以购买新的餐巾，每块餐巾的费用为 P 分；或者把旧餐巾送到快洗部，洗一块需 M 天，其费用为 F 分；或者送到慢洗部，洗一块需 N 天，其费用为 S 分（S< F ）。
- 每天结束时，餐厅必须决定将多少块脏的餐巾送到快洗部，多少块餐巾送到慢洗部，以及多少块保存起来延期送洗。但是每天洗好的餐巾和购买的新餐巾数之和，要满足当天的需求量。
- 试设计一个算法为餐厅合理地安排好 n 天中餐巾使用计划,使总的花费最小。

建模方法

把每天分为二分图两个集合中的顶点Xi,Yi，建立附加源S汇T。

1、从S向每个Xi连一条容量为ri，费用为0的有向边。

2、从每个Yi向T连一条容量为ri，费用为0的有向边。

3、从S向每个Yi连一条容量为无穷大，费用为p的有向边

4、从每个Xi向Xi+1(i+1<=N)连一条容量为无穷大，费用为0的有向边。

5、从每个Xi向Yi+m(i+m<=N)连一条容量为无穷大，费用为f的有向边。

6、从每个Xi向Yi+n(i+n<=N)连一条容量为无穷大，费用为s的有向边。

求网络最小费用最大流，费用流值就是要求的最小总花费。

建模分析

这个问题的主要约束条件是每天的餐巾够用，而餐巾的来源可能是最新购买，也可能是前几天送洗，今天刚刚洗好的餐巾。每天用完的餐巾可以选择送到快洗部或慢洗部，或者留到下一天再处理。

经过分析可以把每天要用的和用完的分离开处理，建模后就是二分图。二分图X集合中顶点Xi表示第i天用完的餐巾，其数量为ri，所以从S向Xi连接容量为ri的边作为限制。Y集合中每个点Yi则是第i天需要的餐巾，数量为ri，与T连接的边容量作为限制。每天用完的餐巾可以选择留到下一天（Xi->Xi+1），不需要花费，送到快洗部(Xi->Yi+m)，费用为f，送到慢洗部(Xi->Yi+n)，费用为s。每天需要的餐巾除了刚刚洗好的餐巾，还可能是新购买的(S->Yi)，费用为p。

在网络上求出的最小费用最大流，满足了问题的约束条件（因为在这个图上最大流一定可以使与T连接的边全部满流，其他边只要有可行流就满足条件），而且还可以保证总费用最小，就是我们的优化目标。

2630 「网络流 24 题」分配问题

习题

- 3409
- 3411
- 3608
- 3947