

线性规划

标准型

描述线性规划问题的常用和最直观形式是标准型。标准型包括以下三个部分：

- 一个需要极大 / 极小化的线性函数：

$$\sum_{i=1}^n x_i$$

- 以下形式的问题约束：

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m \end{aligned}$$

- 和非负变量：

$$x_i \geq 0$$

费用流求解特殊的线性规划问题

先引入一些值恒正的松弛变量，将不等式组转化为等式组。

将每个等式看做一个点，当在特殊情况下如果能构造出“对于所有变量，在等式组中分别在左端和右端出现恰好一次，系数均为 1”的等式组，那么很容易对于每个变量 x 在点（等式）之间进行连边：

例如，等式 p 中， x 出现在等式左端且系数为 1，等式 q 中， x 出现在等式右端且系数为 1，那么连边 $p \rightarrow q$ 流量和费用具体题目进行分析。

另外按需连接源、汇、等式之间通过常数项的连边。最终让连边满足除了源汇，每个点的出流量等于入流量，这样就可以用费用流求出最小或最大费用，而同时这个边的流量便是这个变量的取值。

在特殊的题目中，我们可以加入两个 $0=0$ 的方程并在等式组之间差分，达到这样的要求。

模板题

NEERC 2016 D. Delight for a Cat

题意：有一个人，在某一时刻可以睡觉也可以吃饭，要求连续 k 时刻至少有 m_s 时间在睡觉，至少有 m_e 时刻在吃饭。给定特定时刻睡觉 / 吃饭的快乐值，求最大快乐值以及方案。

题解：先默认他一直吃饭，设 x_i 表示第 i 时刻他是不是在睡觉，很明显 $x_i \in \{0,1\}$ 故关于 x_i 的连边，长度为 1，费用为 $-(s_i - e_i)$ （取负是为了求最小费用）。然后列出不等式组，加入松弛变量转化成等式组：

$$\begin{aligned} 0 &= 0 \\ y_1 + x_1 + x_2 + \cdots + x_k &= k - m_e \\ x_1 + x_2 + \cdots + x_k &= m_s + z_1 \\ y_2 + x_2 + x_3 + \cdots + x_{k+1} &= k - m_e \\ x_2 + x_3 + \cdots + x_{k+1} &= m_s + z_2 \\ &\vdots \\ y_{n-k+1} + x_{n-k+1} + x_{n-k+2} + \cdots + x_n &= k - m_e \\ x_{n-k+1} + x_{n-k+2} + \cdots + x_n &= m_s + z_{n-k+1} \end{aligned}$$

$+x_n \&= m_s + z_{n-k+1} \quad 0 \leq 0$ \end{aligned} \right. \\$. \\$

差分：

$$\begin{aligned} y_1 + x_1 + x_2 + \dots + x_k &= k - m_e \quad k - m_e - m_s \leq y_1 + z_1 \\ x_{k+1} + z_1 + y_2 &= x_1 + k - m_e - m_s \quad k - m_e - m_s \leq y_2 + z_2 \quad \dots \\ x_n + z_1 + y_2 &= x_{n-k} + k - m_e - m_s \quad k - m_e - m_s \leq y_{n-k+1} + z_{n-k+1} \\ m_s + z_{n-k+1} &= x_{n-k+1} + x_{n-k+2} + \dots + x_n \end{aligned} \right. \$. \$$$

连边跑一遍最小费用最大流即可。

```
#include<cstdio>
#include<algorithm>
#include<queue>
#include<map>
#include<cstring>
#include<cmath>
#include<cstdlib>
#include<set>
#include<unordered_map>
#include<vector>
typedef long long ll;
using namespace std;
#define pii pair<int,int>
#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define N 5002
struct Edge{
    int e,n;
    ll l,c;
}e[20*N];
struct Pre{
    int pre,edge;
}pre[N];
int hd[N],vis[N],cnt=1,s,t;
ll maxflow,mincost,dis[N],flow[N];
void add(int a,int b,ll l,ll c){
    e[++cnt].e=b;
    e[cnt].l=l;
    e[cnt].c=c;
    e[cnt].n=hd[a];
    hd[a]=cnt;
}
void add2(int a,int b,ll l,ll c){
    //printf("%d %d %d %d\n",a,b,l,c);
    add(a,b,l,c);
    add(b,a,0,-c);
}
```

```

queue<int>Q;
int spfa(){
    memset(dis,0x7f,sizeof(dis));
    memset(flow,0x7f,sizeof(flow));
    memset(vis,0,sizeof(vis));
    while(!Q.empty())Q.pop();
    int i,top,q;
    Q.push(s);vis[s]=1;dis[s]=0;pre[t].pre=0;
    while(!Q.empty()){
        top=Q.front();
        Q.pop();
        vis[top]=0;
        for(i=hd[top];i;i=e[i].n){
            q=e[i].e;
            if(e[i].l&&dis[q]>dis[top]+e[i].c){
                dis[q]=dis[top]+e[i].c;
                pre[q].pre=top;
                pre[q].edge=i;
                flow[q]=min(flow[top],e[i].l);
                if(!vis[q]){
                    vis[q]=1;
                    Q.push(q);
                }
            }
        }
    }
    return pre[t].pre;
}

void ek(){
    int i;
    while(spfa()){
        maxflow+=flow[t];
        mincost+=flow[t]*dis[t];
        for(i=t;i!=s;i=pre[i].pre){
            e[pre[i].edge].l-=flow[t];
            e[pre[i].edge^1].l+=flow[t];
        }
    }
}

int val_s[N],val_e[N],edge_id[N];
int main(){
    int i,n,k,ms,me;
    ll ans=0;
    freopen("delight.in","r",stdin);
    freopen("delight.out","w",stdout);
    scanf("%d%d%d%d",&n,&k,&ms,&me);
    s=2*(n-k+1)+2;t=s+1;
    for(i=1;i<=n;i++)scanf("%d",&val_s[i]);
    for(i=1;i<=n;i++)scanf("%d",&val_e[i]),ans+=val_e[i];
    // x_i
    for(i=1;i<=n;i++){

```

```
int l=(i<=k)?1:2*(i-k)+1;
int r=(i>=n-k+1)?2*(n-k+1)+1:1+2*i;
add2(l,r,1,-(val_s[i]-val_e[i]));
edge_id[i]=cnt-1;
}
// y_i, z_i
for(i=1;i<=n-k+1;i++){
    add2(2*i-1,2*i,1e9,0); // y_i
    add2(2*i+1,2*i,1e9,0); // z_i
}
// constant
add2(s,1,k-me,0);
for(i=2;i<=2*(n-k+1);i++){
    if(i%2==0)add2(i,t,k-me-ms,0);
    else add2(s,i,k-me-ms,0);
}
add2(2*(n-k+1)+1,t,ms,0);
ek();
printf("%lld\n",-mincost+ans);
for(i=1;i<=n;i++)printf("%c",e[edge_id[i]].l?'E':'S');
return 0;
}
```

□□

练习题

NOI 2008 志愿者招募 题解

题意：有 n 天， m 种工作时间为 $[s_i, t_i]$ 工资为 v_i 的志愿者，每个时间点有志愿者个数要求，求最小花费。

题解：抽象出来每个方程的样子，发现对于每个 x_i 差分之后连的是一条 $s_i \rightarrow t_i + 1$ 的边，松弛变量和常数与上题类似。

```
#include<cstdio>
#include<algorithm>
#include<queue>
#include<map>
#include<cstring>
#include<cmath>
#include<cstdlib>
#include<set>
#include<unordered_map>
#include<vector>
typedef long long ll;
using namespace std;
#define pii pair<int,int>
```

```

#define pb push_back
#define mp make_pair
#define fi first
#define se second
#define N 5002
struct Edge{
    int e,n;
    ll l,c;
}e[20*N];
struct Pre{
    int pre,edge;
}pre[N];
int hd[N],vis[N],cnt=1,s,t;
ll maxflow,mincost,dis[N],flow[N];
void add(int a,int b,ll l,ll c){
    e[++cnt].e=b;
    e[cnt].l=l;
    e[cnt].c=c;
    e[cnt].n=hd[a];
    hd[a]=cnt;
}
void add2(int a,int b,ll l,ll c){
    //printf("%d %d %d %d\n",a,b,l,c);
    add(a,b,l,c);
    add(b,a,0,-c);
}
queue<int>Q;
int spfa(){
    memset(dis,0x7f,sizeof(dis));
    memset(flow,0x7f,sizeof(flow));
    memset(vis,0,sizeof(vis));
    while(!Q.empty())Q.pop();
    int i,top,q;
    Q.push(s);vis[s]=1;dis[s]=0;pre[t].pre=0;
    while(!Q.empty()){
        top=Q.front();
        Q.pop();
        vis[top]=0;
        for(i=hd[top];i;i=e[i].n){
            q=e[i].e;
            if(e[i].l&&dis[q]>dis[top]+e[i].c){
                dis[q]=dis[top]+e[i].c;
                pre[q].pre=top;
                pre[q].edge=i;
                flow[q]=min(flow[top],e[i].l);
                if(!vis[q]){
                    vis[q]=1;
                    Q.push(q);
                }
            }
        }
    }
}

```

```
}
return pre[t].pre;
}
void ek(){
    int i;
    while(spfa()){
        maxflow+=flow[t];
        mincost+=flow[t]*dis[t];
        for(i=t;i!=s;i=pre[i].pre){
            e[pre[i].edge].l-=flow[t];
            e[pre[i].edge^1].l+=flow[t];
        }
    }
}
int a[N];
int main(){
    int i,n,m,ms,me;
    scanf("%d%d",&n,&m);
    for(i=1;i<=n;i++)scanf("%d",&a[i]);
    s=n+2;t=s+1;
    for(i=0;i<m;i++){
        int l,r,v;
        scanf("%d%d%d",&l,&r,&v);
        add2(l,r+1,1e18,v);
    }
    for(i=1;i<=n;i++)add2(i+1,i,1e18,0);
    add2(s,1,a[1],0);
    for(i=2;i<=n;i++){
        if(a[i]-a[i-1]>=0)add2(s,i,a[i]-a[i-1],0);
        else add2(i,t,a[i-1]-a[i],0);
    }
    add2(n+1,t,a[n],0);
    ek();
    printf("%lld\n",mincost);
    return 0;
}
```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:i_dont_know_png:potassium:linear_programming

Last update: 2020/05/24 19:59