差分约束系统

差分约束系统是一个特殊的n元一次不等式组，每项不等式约束条件都可以写成$x_i \leq x_j + c_k$的形式，这与单源最短路的三角形不等式类似，因此可以将每个变量x_i看成图的一个顶点，边权代表两变量之间的大小关系来建立模型，通过跑最短/长路，可以解决该不等式组的最小/最大解的问题。

在实际问题中，我们会遇到建图存在负环，即解不存在的情况。我们通常可以用Bellman-Ford算法和SPFA算法来解决。差分约束的关键在于建图，需要对建立模型并进行分析。之后就是简单的跑最短/长路了。

常用技巧

- 对于不等式组，我们可以乘一个-1来使得规范不等号的方向是同一侧的。
- 对于多个变量的不等式，我们往往需要借助其他不等式来简化成两变量间的关系。
- 考虑用栈代替队列来过卡SPFA的数据。
- 一般连边形式：$x_b - x_a \leq c \Rightarrow addedge(a, b, c)$

例题

luogu P1993 小 K 的农场

题目大意：求解差分约束系统，有m条约束条件，每条都为形如$x_a - x_b \geq c_k$， 或$x_a = x_b$的形式，判断该差分约束系统有没有解。

纯模板，按技巧4常规建图即可，可以从一个超级源点出发。

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
using namespace std;
const int N=10007;
const int mod=1e9+7;

struct E{int v,w,nxt;}e[N<<1];
int head[N],cnt=0;
inline void addedge(int u,int v,int w){e[++cnt]=(E){v,w,head[u]};head[u]=cnt;}

int n,m,dis[N],inq[N],num[N];

bool spfa(){
```

```
    queue<int>q;
    memset(dis,inf,sizeof(dis));
    dis[0]=0;
    q.push(0);
    while(q.size()){
        int fro=q.front();
        q.pop();
        inq[fro]=0;
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v;
            if(dis[to]>dis[fro]+e[i].w){
                dis[to]=dis[fro]+e[i].w;
                num[to]=num[fro]+1;
                if(num[to]>n) return false;
                if(!inq[to]){
                    inq[to]=1;
                    q.push(to);
                }
            }
        }
    }
    return true;
}

void Solve(){
    cin>>n>>m;
    for(int i=1,op,a,b,c;i<=m;++i){
        cin>>op>>a>>b;
        if(op==1){
            cin>>c; //xb-xa <=-c
            addedge(a,b,-c);
        }else if(op==2){
            cin>>c;
            addedge(b,a,c); //xa-xb <=c
        }else{
            addedge(a,b,0);
            addedge(b,a,0);
        }
    }
    for(int i=1;i<=n;++i) addedge(0,i,0);
    if(spfa()) cout<<"Yes\n";
    else cout<<"No\n";
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    int T=1;
    while(T--){
        Solve();
    }
}
```

「SCOI2011」糖果

将糖果分给n个小朋友，需要满足的k个关系，每行关系用3个数字x,a,b表示。
如果x=1,表示第a个小朋友分到的糖果必须和第b个小朋友分到的糖果一样多。
如果x=2，表示第a个小朋友分到的糖果必须少于第b个小朋友分到的糖果。
如果x=3，表示第a个小朋友分到的糖果必须不少于第b个小朋友分到的糖果。
如果x=4，表示第a个小朋友分到的糖果必须多于第b个小朋友分到的糖果。
如果x=5，表示第a个小朋友分到的糖果必须不多于第b个小朋友分到的糖果。

问至少需要准备多少糖果。

也是按上面的不等式来建图，差分约束跑最长路就出来了。注意我这里用了个栈，写成队列被卡了。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+7,M=3e5+7;
typedef long long ll;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(ll x){if(x>9) write(x/10);putchar(x%10+'0');}
struct E{
    int v,w,nxt;
}e[M];

int head[N],cnt=0;
int n,k,num[N];
bool inq[N];
ll dis[N],ans=0;
stack<int>st;

void addedge(int u,int v,int w){
    e[++cnt]=(E){v,w,head[u]};head[u]=cnt;
}

signed main(){
    n=read();k=read();
    for(int i=1,op,u,v;i<=k;++i){
        op=read();u=read();v=read();
        if(op==1) addedge(u,v,0),addedge(v,u,0);
        else if(op==2) addedge(u,v,1);
        else if(op==3) addedge(v,u,0);
        else if(op==4) addedge(v,u,1);
        else if(op==5) addedge(u,v,0);
    }
    for(int i=1;i<=n;++i) addedge(0,i,1); //建一个超级源点
    st.push(0);dis[0]=0;
    while(st.size()){
        int fro=st.top();
        st.pop();
        inq[fro]=0;
        for(int i=head[fro],to;i;i=e[i].nxt){
            to=e[i].v;
            if(dis[to]<dis[fro]+e[i].w){ //最长路
                dis[to]=dis[fro]+e[i].w;
                num[to]=num[fro]+1;
                if(num[to]>n){ //判环
```

```
                    puts("-1");
                    return 0;
                }
                if(!inq[to]){
                    inq[to]=1;
                    st.push(to);
                }
            }
        }
    }
    for(int i=1;i<=n;++i) ans+=dis[i];
    write(ans);
    return 0;
}
```

Intervals

给定n个区间以及这个区间最少包含选中的数的个数，问最少需要选中多少个数。

将整个区间的所有数都看作结点，那么我们根据A到B点最少需要选中多少数来进行建边，注意两点之间有两条不等式，所以再建一组反向边反向边。

$$\begin{cases} x_i - x_{i-1} \leq 1 \\ x_i - x_{i-1} \geq 0 \end{cases}$$

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
using namespace std;
const int N=5e5+7;

struct E{int v,w,nxt;}e[N<<1];
int head[N],cnt=0;
void addedge(int u,int v,int w){
    e[++cnt]=(E){v,w,head[u]};head[u]=cnt;
}

int n,l,r,dis[N],inq[N];
queue<int>q;

signed main(){
    scanf("%d",&n);
    for(int i=1,u,v,w;i<=n;i++){ //差分约束系统建图
        scanf("%d%d%d",&u,&v,&w);
        addedge(u-1,v,w);
        l=min(l,u-1);
        r=max(r,v);
    }
    for(int i=l;i<=r;++i){
        addedge(i,i+1,0); //i+1不放数
        addedge(i+1,i,-1); //退一个数
        dis[i]=-inf;
    }
    dis[l]=0;inq[l]=1;
    q.push(l);
    while(!q.empty()){
        int fro=q.front();
```

```
        q.pop();
        inq[fro]=0;
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v;
            if(dis[to]<dis[fro]+e[i].w){ //跑最长路
                dis[to]=dis[fro]+e[i].w;
                if(!inq[to]){
                    q.push(to);
                    inq [to]=1;
                }
            }
        }
    }
    printf("%d",dis[r]);
    return 0;
}
```

Cashier Employment（出纳员问题）

给定24小时中每小时需要的职员数，以及n个职员的开始工作时间，每个职员都可以连续工作8小时，问最少需要选多少职员才可以满足需要?

先假定k个员工可以满足，那么我们可以对每个时刻进行建边（在该节点放一个员工）之后，肯定是满足最长路dis[0->23]==k的。

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<cstring>
#define inf 0x3f3f3f3f

using namespace std;
const int N=30,M=2e5+7;
const int mod=1e9+7;

struct E{int v,w,nxt;}e[M];
int head[N],cnt;
inline void addedge(int u,int v,int w){e[++cnt]=(E){v,w,head[u]};head[u]=cnt;}

int n,need[N],work[N],dis[N],inq[N],used[N];

inline int spfa(int k){
    cnt=0;
    memset(head,0,sizeof(head));
    for(int i=1;i<=24;++i){
        addedge(i-1,i,0);
        addedge(i,i-1,-work[i]);
        if(i>=8) addedge(i-8,i,need[i]);
        else addedge(i+16,i,need[i]-k);
    }
    addedge(0,24,k); //24小时内有k个人 x
    for(int i=0;i<=24;++i) dis[i]=-inf,inq[i]=0,used[i]=0;
    dis[0]=0;inq[0]=1;used[0]=1;
    queue<int>q;
    q.push(0);
```

```
    while(q.size()){
        int fro=q.front();
        q.pop();
        inq[fro]=0;
        for(int i=head[fro];i;i=e[i].nxt){
            int v=e[i].v,w=e[i].w;
            if(dis[v]<dis[fro]+w){
                dis[v]=dis[fro]+w;
                if(!inq[v]){
                    inq[v]=1;
                    ++used[v];
                    if(used[v]>24) return 0;//存在负环
                    q.push(v);
                }
            }
        }
    }
    return dis[24]==k;
}

void Solve(){
    memset(need,0,sizeof need);
    memset(work,0,sizeof work);
    for(int i=1;i<=24;++i){
        cin>>need[i];
        addedge(i-1,i,need[i]);
    }
    cin>>n;
    for(int i=1,s;i<=n;++i){
        cin>>s;
        ++work[s+1];
    }
    int L=0,R=n+1,mid;
    while(R-L>1){  //二分以下需要的人数，直接枚举也可以
        mid=((L+R)>>1);
        if(spfa(mid)) R=mid;
        else L=mid;
    }
    if(spfa(L)) R=L;
    if(R>n) cout<<"No Solution\n";
    else cout<<R<<"\n";
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    int T=1;
    cin>>T;
    while(T--){
        Solve();
    }
}
```

luoguP4926 [1007]倍杀测量者

如果某选手不满足以下限制条件，他将会被强迫女装

- $s[A] > s[B] * k_i$
- $s[A] > s[B] * \frac{1}{k_i}$

其中s[A]表示选手A的分数。同时你还已知一些选手的分数

请你求出最大的正实数T，使得限制条件变成下面的形式后，仍然有人女装

- $s[A] > s[B] * (k_i - T)$
- $s[A] > s[B] * \frac{1}{k_i+T}$

```
#include<bits/stdc++.h>
#define eps 1e-8
using namespace std;
const int N=1e3+7;
struct Q{int o,a,b,k;}fl[N];
struct E{int nxt,to;double w;}e[N<<1];
int n,s,t,c[N],num[N],head[N],cnt,inq[N];
double dis[N];queue<int>q;
void addedge(int u,int v,double w){e[++cnt]=(E){head[u],v,w};head[u]=cnt;}
int check(double T){
    memset(head,0,sizeof(head));
    memset(num,0,sizeof(num));
    cnt=0;
    while(q.size()) q.pop();
    for(int i=0;i<=n;i++) dis[i]=1,inq[i]=1,q.push(i);
    for(int i=1;i<=n;i++)
        if(c[i]) addedge(i,0,1.0/c[i]),addedge(0,i,c[i]);
    for(int i=1;i<=s;i++){
        int A=fl[i].a,B=fl[i].b,k=fl[i].k,o=fl[i].o;
        if(o==1) addedge(B,A,k-T);
        else addedge(B,A,1.0/(k+T));
    }
    while(!q.empty()){
        int fro=q.front();
        q.pop();
        inq[fro]=0;
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].to;
            if(dis[to]>=dis[fro]*e[i].w) continue;
            dis[to]=dis[fro]*e[i].w;
            num[to]=num[fro]+1;
            if(num[to]>n) return 1;
            if(!inq[to]) q.push(to),inq[to]=1;
        }
    }
    return 0;
}

signed main(){
    cin>>n>>s>>t;
    double l=0,r=1e18,T=-1;
    for(int i=1,op,a,b,k;i<=s;i++){
        cin>>op>>a>>b>>k;
        fl[i]=(Q){op,a,b,k};
```

```
        if(op==1) r=min(r,(double)k-eps);
    }
    for(int i=1,C,x;i<=t;i++) cin>>C>>x,c[C]=x;
    while(r-l>eps){
        double mid=(l+r)/2;
        check(mid)?l=T=mid:r=mid;
    }
    T==-1?puts("-1"):printf("%.10lf\n",T);
}
```

「一本通 3.4 练习 2」布局 Layout

给定n头牛要求按顺序排队，$m1$个关系(u,v,w)表示u和v的距离要小于等于w，$m2$个关系(u,v,w)表示u和v的距离要大于等于w。如果没有合法方案，输出 -1. 如果有合法方案，但 1 号奶牛可以与 N 号奶牛相距无穷远，输出 -2. 否则，输出 1 号奶牛与 N 号奶牛间的最大距离。

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
using namespace std;
const int N=1e3+7,M=3e4+7;

struct E{int v,w,nxt;}e[M];
int cnt=0,head[N];
inline void addedge(int u,int v,int w){e[++cnt]=(E){v,w,head[u]};head[u]=cnt;}

int n,m1,m2,dis[N],vis[N],num[N];
inline bool spfa(int sz){
    for(int i=0;i<=n;++i) dis[i]=inf,vis[i]=0,num[i]=0;
    queue<int>q;
    for(int i=1;i<=sz;++i) q.emplace(i),dis[i]=0,vis[i]=1;
    while(q.size()){
        int fro=q.front();
        q.pop();
        vis[fro]=0;
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v,w=e[i].w;
            if(dis[to]>dis[fro]+w){
                dis[to]=dis[fro]+w;
                ++num[to];
                if(num[to]>=n) return true; //有环无方案
                if(!vis[to]){
                    vis[to]=1;
                    q.emplace(to);
                }
            }
        }
    }
    return false;
}

signed  main(){
    scanf("%d%d%d",&n,&m1,&m2);
    for(int i=1,u,v,w;i<=m1;++i){
        scanf("%d%d%d",&u,&v,&w);
        addedge(u,v,w);
```

```
    }
    for(int i=1,u,v,w;i<=m2;++i){
        scanf("%d%d%d",&u,&v,&w);
        addedge(v,u,-w);
    }
    for(int i=1;i<n;++i) addedge(i,i-1,0);
    if(spfa(n)) puts("-1");
    else{
        spfa(1);
        if(dis[n]>=inf) puts("-2");
        else printf("%d\n",dis[n]);
    }
    return 0;
}
```

参考资料

OI-Wiki

《差分约束》 胡泽聪

NOIP知识点串讲——Colin

最后一题题解https://www.luogu.com.cn/blog/xzyxzy/solution-p4926