前情提要

给定n个点，m条边，求两点(u,v)之间的最短距离

图上跑最短路求解

改一下

给定n个点，m条边，允许你操作k条边（比如花费变为0），
求两点(u,v)之间的最短距离

DP解决

f [v][j]——走到v免费j条路径的最小花费。

状态转移方程：

f [u][j]=min(f [v][j]+w(u,v),f [v][j+1])，存在边u->v

信息学

图论-分层图思想

西南大学附属中学校

信息奥赛教练组

什么是分层图

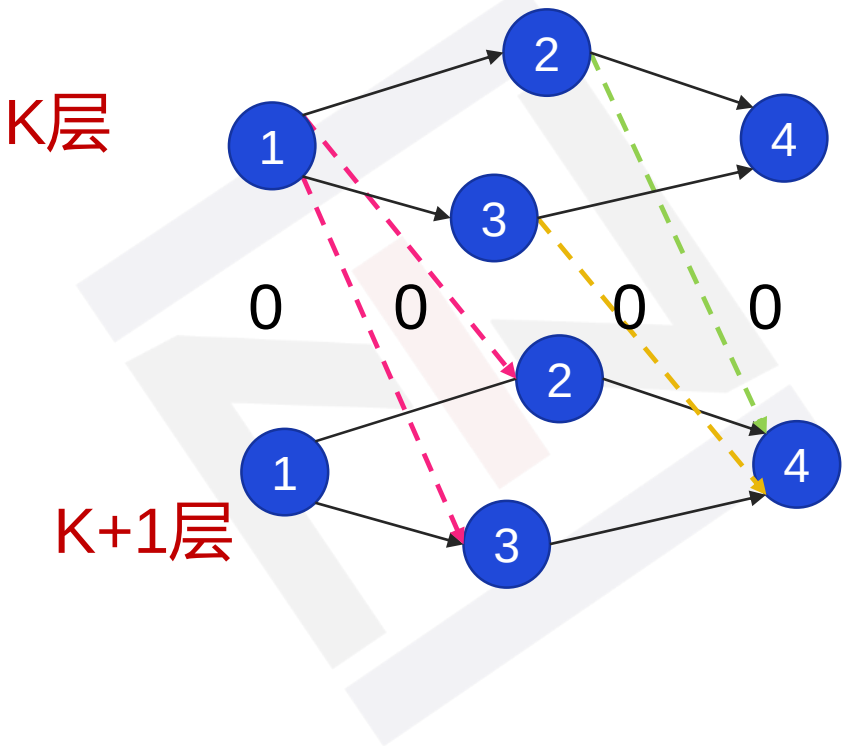
就是分成很多层的图　　（这是什么废话）

分层图是一种建模思想，并不是一种算法

怎么建模

如果有 k 次修改机会，那么建立 k+1 层

什么是分层图



建立多层相同或相似的图，并在图与图之间进行连边，可以实现两种性质的图之间的转移。

一般将决策前的状态和决策后的状态之间连接一条权值为决策代价的边，表示付出该代价后就可以转换状态了。

分层图的构建步骤可以描述为：

1、先将图复制成 k+1 份 （0 ~ k）

2、对于图中的每一条边 <u,v> 从 u_i 到 v_{i+1} 建立与题目所给操作相对应的边（i=0,1,…,k）

K层图如何保存?

1：物理建图：实实在在的建图，n 层

一般是建一维的。
比如一张图有 n 个点，我们要建 k+1 张图，那么用一个 n*k 的数列就可以表示了，其中对于第 i 张图的起点就应该是（i-1）*n，因为 0-n 的区间也是要拿来建图的。

```
add(u,v,w);
add(v,u,w);
//添加边的时候添加其他层
for (int j=1;j<=k;j++){
  add(u+j*n-n,v+j*n,0);
  add(v+j *n-n,u+j*n,0);
  add(u+j*n,v+j*n,w);
  add(v+j*n,u+j*n,w);
}
```

注意

这种方法因为要建多张图，
所以可能比较耗内存，就很容易被卡死。

K层图如何保存?

2：逻辑建图：逻辑上的对图分层，一般就是给dis数组或者vis数组，

dis[i] [j] 代表到达 i 用了 j 次免费机会的最小花费

vis[i] [j] 代表到达 i 用了 j 次免费机会的情况是否出现过

方法：给需要参与求解实际问题的数据结构额外增加一维数组来模拟n层的效果

```
struct node
{
    int tier; //层数
    int d; //到达该点最短路
    int num; //当前结点
}a[k*n]; //kn 个点
```

「BJWC2012」冻结

西南大学附属中学 High School Affiliated to Southwest University

题目描述

我们考虑最简单的旅行问题吧： 现在这个大陆上有 N 个城市，M 条双向的道路。城市编号为 $1 \sim N$，我们在 1 号城市，需要到 N 号城市，怎样才能最快地到达呢?

这不就是最短路问题吗？我们都知道可以用 Dijkstra、Bellman-Ford、Floyd-Warshall等算法来解决。

现在，我们一共有 K 张可以使时间变慢 50%的 SpellCard，也就是说，在通过某条路径时，我们可以选择使用一张卡片，这样，我们通过这一条道路的时间 就可以减少到原先的一半。需要注意的是：

1. 在一条道路上最多只能使用一张 SpellCard。
2. 使用一张SpellCard 只在一条道路上起作用。
3. 你不必使用完所有的 SpellCard。

给定以上的信息，你的任务是：求出在可以使用这不超过 K 张时间减速的 SpellCard 之情形下，从城市 1 到城市 N 最少需要多长时间。

输入输出格式

输入格式

第一行包含三个整数：N、M、K。

接下来 M 行，每行包含三个整数：A_i、B_i、$Time_i$，表示存在一条 A_i 与 B_i 之间的双向道路，在不使用 SpellCard 之前提下，通过它需要 $Time_i$ 的时间。

输出格式

输出一个整数，表示从 1 号城市到 N 号城市的最小用时。

法一：物理建图

西南大学附属中学 High School Affiliated to Southwest University

```
const int maxn=55,maxk=55,maxe=1010;
const int inf=0x3f3f3f3f;

int head[maxn*maxk],cnt=0;

struct edge{
    int v,next;
    int w;
}e[maxe*maxk*4];

void add(int u,int v,int w){//添加边
    e[cnt].v=v;
    e[cnt].w=w;
    e[cnt].next=head[u];
    head[u]=cnt++;
}
```

```
int n,m,k;
int dis[maxn*maxk];
bool vis[maxn*maxk];

void dj(int s){
    memset(vis,0,sizeof(vis));
    memset(dis,inf,sizeof(dis));
    dis[s]=0;
    //堆优化  小根堆
    priority_queue<pair<int,int>,vector< pair<int,int> >,greater< pair<int,int> > > q;
    q.push(make_pair(0,s)) ;

    while(!q.empty()){
        int u=q.top().second;
        q.pop();

        if(vis[u]) continue;
        vis[u]=1;
        for(int i=head[u];~i;i=e[i].next){
            int v=e[i].v;
            if(dis[v]>dis[u]+e[i].w){
                dis[v]=dis[u]+e[i].w;
                q.push(make_pair(dis[v],v));
            }
        }

    }
}
```

法一：物理建图


```
int main(){
    //初始化头
    memset(head,-1,sizeof(head));
    scanf("%d%d%d",&n,&m,&k);
    for(int i=1;i<=m;i++){
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        //第0层
        add(u,v,w);
        add(v,u,w);
        //建立k层
        for(int j=1;j<=k;j++){
            add(u+j*n-n,v+j*n,w/2);
            add(v+j*n-n,u+j*n,w/2);
            add(u+j*n,v+j*n,w);
            add(v+j*n,u+j*n,w);
        }
    }
        dj(1);
        //循环一遍看不同操作次数的最小值
        int ans=inf;
        for (int i=0;i<=k;i++){
            ans=min(ans,dis[n+i*n]);
        }
        cout<<ans<<endl;

    return 0;
}
```

法二：逻辑建图


```
const int maxn=55,maxk=55;
int n, m, k, ans = 0x7f3f3f3f, cnt, a, b, c;
int dis[maxn][maxk], vis[maxn][maxk], first[maxn];
int next[2010], v[2010], w[2010];
struct node {   //第tot层的结点u，最短路为val
    int val;
    int u;
    int tot;
};
bool operator<(node a, node b)//重载运算符，小根堆写法
{
    return a.val > b.val;
}
priority_queue<node> q;

void add(int U, int V, int W)
{
    v[++cnt] = V;
    w[cnt] = W;
    next[cnt] = first[U];
    first[U] = cnt;
}
```

法二：逻辑建图


```cpp
int main()
{
    scanf("%d%d%d",&n,&m,&k);
    //建图
    for (int i = 1; i <= m; i++) {
        cin >> a >> b >> c;
        add(a, b, c);
        add(b, a, c);
    }
    memset(dis, 0x3f, sizeof(dis));

    dis[1][0] = 0;
    q.push({ 0, 1, 0 });
    while (q.size()) {
        int u = q.top().u, tot = q.top().tot;
        q.pop();
        if (vis[u][tot])
            continue;
        vis[u][tot] = 1;
        for (int i = first[u]; i; i = next[i]) {
            if (dis[v[i]][tot] > dis[u][tot] + w[i]) {//不免费在同一层
                dis[v[i]][tot] = dis[u][tot] + w[i];
                q.push({ dis[v[i]][tot], v[i], tot });
            }
            if (tot < k && dis[v[i]][tot + 1] > dis[u][tot] + w[i] / 2) {//免费，在下一层
                dis[v[i]][tot + 1] = dis[u][tot] + w[i] / 2;
                q.push({ dis[v[i]][tot + 1], v[i], tot + 1 });
            }
        }
    }
    //循环一遍找最小值
    for (int i = 0; i <= k; i++)
        ans = min(ans, dis[n][i]);
    cout << ans;
    return 0;
}
```

小结一下

1.分层图是一种建模思想，不是算法。

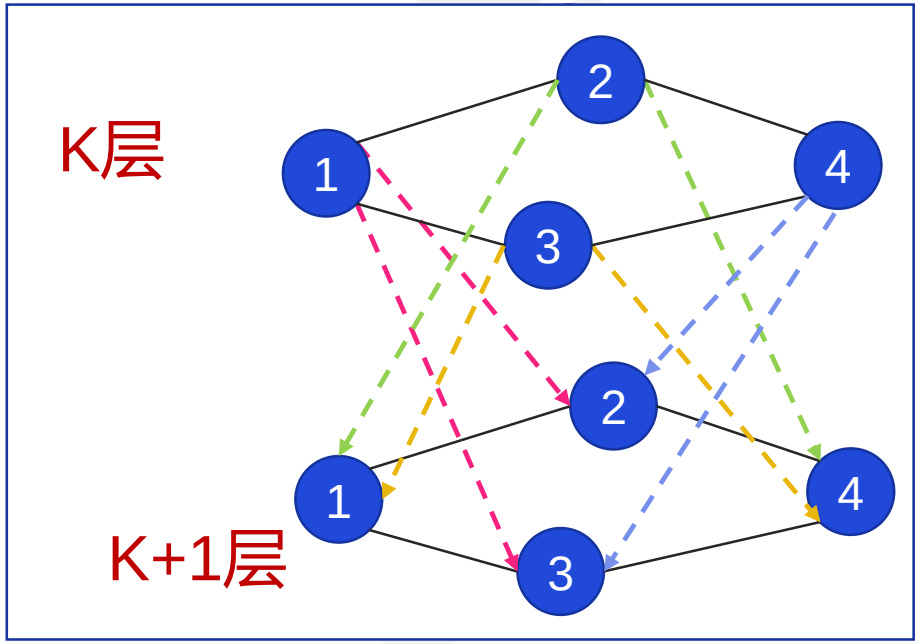
2.分层图和最短路搭配使用。

3.分层图适用于对边有干扰操作（如能够将边权降低）。我们对这些干扰操作的解决方 法就是把原图“复制”，并且一般来说干扰操作有多少次就要复制多少。但是这些操作的数目不太大。

4.DP的写法与分层图的一模一样，只是DP中没有层的概念，表示的是使用的免费次数，两者是不同的思想

空间复杂度

M条边 n个点 k次机会



有向图 **(2k+1)m**

每层边的数量 m

一共有多少层 K+1 原始图有一层

层间有多少边 m

一共有多少层 K

无向图 **(4k+1)m**

每层边的数量 m *2 储存时两个方向都存

一共有多少层 K+1

层间有多少边 2*m

一共有多少层 K

```
 for (int i = first[u]; i; i = next[i]) {
      if (dis[v[i]] > dis[u] + w[i]) {
          dis[v[i]] = dis[u] + w[i];
          q.push({ dis[v[i]],v[i], tot });
       }
}
```

```
 for (int i = first[u]; i; i = next[i]) {
      if (dis[v[i]][tot] > dis[u][tot] + w[i]) {//不免费在同一层
          dis[v[i]][tot] = dis[u][tot] + w[i];
          q.push({ dis[v[i]][tot], v[i], tot });
       }
      if (tot < k && dis[v[i]][tot + 1] > dis[u][tot] + w[i] / 2) {//免费，在下一层
          dis[v[i]][tot + 1] = dis[u][tot] + w[i] / 2;
          q.push({ dis[v[i]][tot + 1], v[i], tot + 1 });
       }
 }
```

一个问题

已知下列不等式方程组,问 $x_3 - x_1$ 的最大值

$$\begin{aligned} x_2 - x_1 &\leq C_1 \\ x_3 - x_2 &\leq C_2 \\ x_3 - x_1 &\leq C_3 \end{aligned}$$

1：**数学方法**合并前两个不等式可以得到

$$\begin{aligned} x_3 - x_1 &\leq C_1 + C_2 \\ x_3 - x_1 &\leq C_3 \end{aligned}$$

→ $x_3 - x_1$ 的最大值 $min(C_1 + C_2, C_3)$

一个问题

已知下列不等式方程组，问 $x_3 - x_1$ 的最大值

$$x_2 - x_1 \leq C_1$$
$$x_3 - x_2 \leq C_2$$
$$x_3 - x_1 \leq C_3$$

2：图论方法

$x_i <= x_j + c$

dis [j] <= dis [i] +w[i][j]

最短路

如果图中不存在负权回路，则当最短路算法结束以后，对于边< i , j >，有dis [j] <= dis [i] +w[i][j] ，即 dis [j] - dis [i] <= w[i][j] 成立

```
void spfa (int u)
{
    int v, w, i;
    for (i = 1; i <= n; i++)    //对于从1到n的编号
        dist[i] = inf, inq[i] = false;
    dist[u] = 0;
    queue<int> q;
    q.push (u);
    inq[u] = true;
    while (!q.empty())
    {
        u = q.front();
        q.pop();
        inq[u] = false;
        for (i = pre[u]; i != -1; i = e[i].next)
        {
            w = e[i].w;
            v = e[i].v;
            if (dist[u] + w < dist[v])
            {
                dist[v] = dist[u] + w;
                if (!inq[v])
                {
                    q.push (v);
                    inq[v] = true;
                }
```

信息学

图论-差分约束系统

西南大学附属中学校

信息奥赛教练组

什么是差分约束系统

差分约束系统是一种特殊的N元一次不等式组，包含N个变量X1-XN以及M个约束条件，每个约束条件是由两个变量作差构成的，如Xi-Xj<=Ck

解决的问题

1. 是否有可行解？

$$\begin{cases} x_1 - x_2 \le 0 \\ x_1 - x_5 \le 1 \\ x_2 - x_5 \le 1 \\ x_3 - x_1 \le 5 \\ x_4 - x_1 \le 4 \\ x_4 - x_3 \le -1 \\ x_5 - x_3 \le -3 \\ x_5 - x_4 \le -3 \end{cases}$$

求出一组解

问题1：是否有解

$$\begin{cases} x_1 - x_2 \le 0 \\ x_1 - x_5 \le 1 \\ x_2 - x_5 \le 1 \\ x_3 - x_1 \le 5 \\ x_4 - x_1 \le 4 \\ x_4 - x_3 \le -1 \\ x_5 - x_3 \le -3 \\ x_5 - x_4 \le -3 \end{cases}$$

建图

Xj -Xi <= C　　dis [j] - dis [i] <= w[i][j]

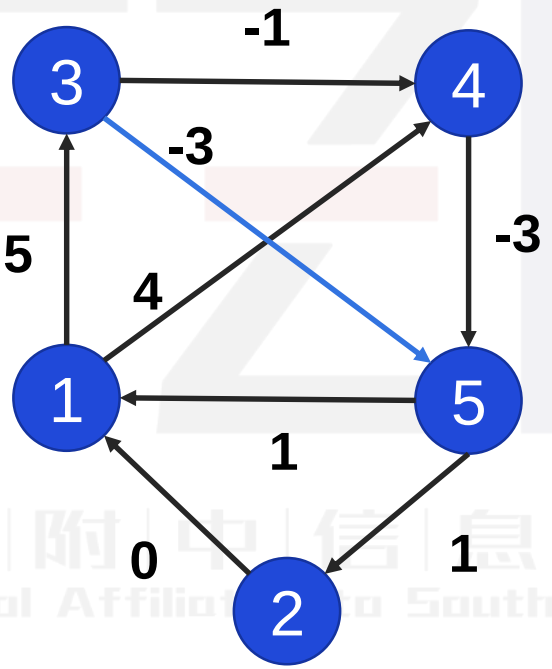
有一条**从 i 指向 j** 的边，**边权为 c**

将x1,x2等当作到图中的点的最短路长度，
约束关系作为边权

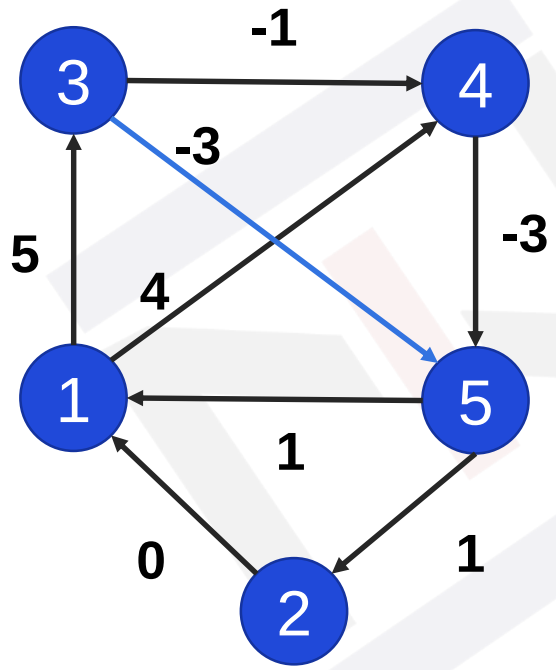
问题1：是否有解

$$\begin{cases} x_1 - x_2 \le 0 \\ x_1 - x_5 \le 1 \\ x_2 - x_5 \le 1 \\ x_3 - x_1 \le 5 \\ x_4 - x_1 \le 4 \\ x_4 - x_3 \le -1 \\ x_5 - x_3 \le -3 \\ x_5 - x_4 \le -3 \end{cases}$$

建图



问题1：是否有解



怎么判断是否有解?

X代表到该点的**最短路长度**

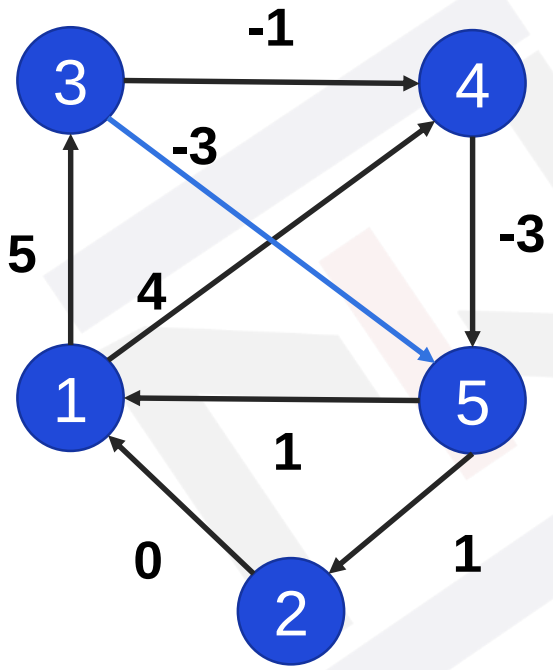
（1）无解：存在负权回路（Ford，SPFA）

（2）有解：

不存在负权回路，求出的源点到每个点的最短路dis就是一组解

哪一个点是源点?

哪一个点是源点?



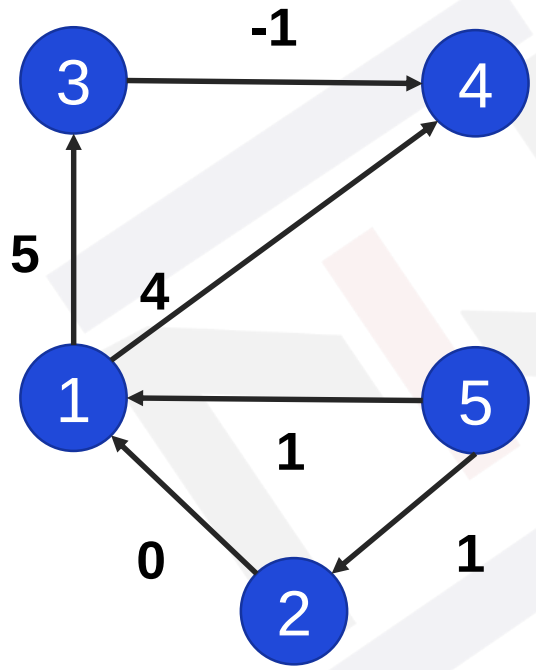
以1为起点，则起点到各个顶点的最短距离为
dis[1]=0,dis[2]=2,dis[3]=5,dis[4]=4,dis[5]=1，
则得到解 x1=0,x2=2,x3=5,x4=4,x5=1

以3为起点，则起点到各个顶点的最短距离为
dis[1]=−3,dis[2]=−-3,dis[3]=0,dis[4]=−1,dis[5]=−4，
则得到解 x1=2,x2=2,x3=5,x4=4,x5=1

以不同顶点作为起点会得到不同的解，但这些解都是合理

所有点都可以?

所有点都可以?



1 与 5 不连通，从 1 出发没办法得到5的最短路

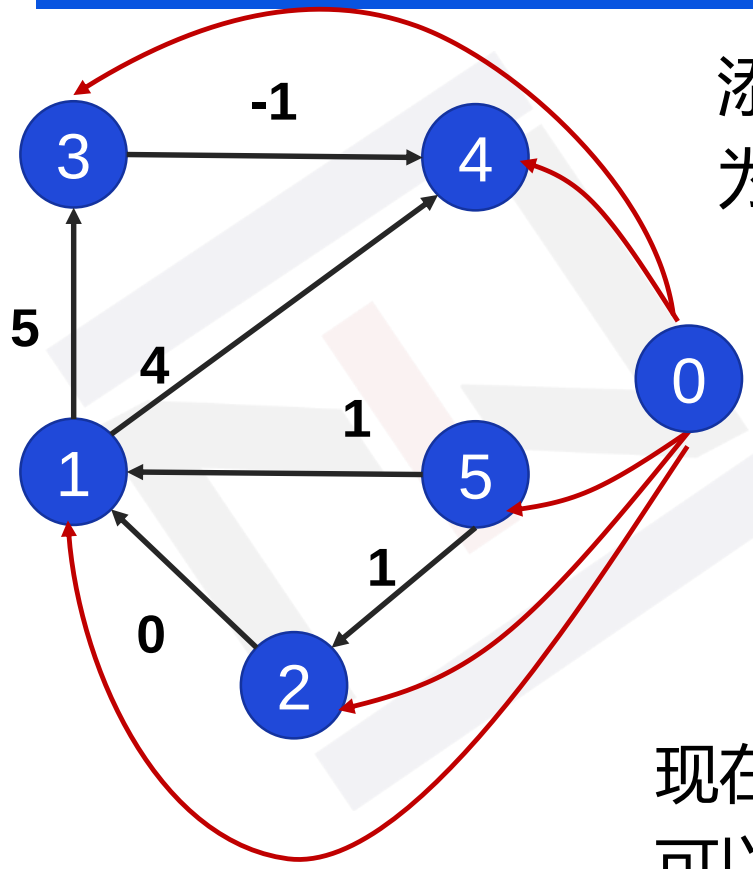
有时候图可能**不是强连通的**，

选择图中的点作为起点不一定能求出到其他点的最短路

怎么解决?

添加一个超级源点X_0，从X_0与每个顶点都连接一条权值为0的边，相当于添加了一个不等式组

怎么解决?



添加一个超级源点X_0，从X_0与每个顶点都连接一条权值为0的边，相当于添加了一个不等式组

$$X1-x0<=0$$
$$X2-x0<=0$$
$$X3-x0<=0$$
$$X4-x0<=0$$
$$X5-x0<=0$$

现在**以X_0为起点**，求出到Xi 的最短路就是Xi 的解，可以根据题目要求，可以整体增加 d (d为常数)。

差分约束系统能解决什么问题

求解两者相差最大多少?

求解A-B的最大值，即A-B≤?

通过形如 Xi-Xj≤D的不等式，建图，存在 Xi->Xj 的边，边权为D。

求解B到A的最短路(顺序不能颠倒)。

求解两者相差最小多少?

求解A-B的最小值，即A-B≥?

通过形如 Xi-Xj≥D的不等式，建图，存在 Xi->Xj 的边，边权为D。

求解B到A的最长路(顺序不能颠倒)

两种处理方法

对于不等式形如： **Xi-Xj>=Ck**

（1）可以转换成**Xi>=Xj+Ck**，求最长路。

（2）两边同时乘以-1，转换成**Xj-Xi<=-Ck**，求最短路

如果存在形如：**Xi-Xj<Ck**

可以转换成 **Xi-Xj<=Ck-1**，解为整数