图论-差分约束

一、AcWing 1169. 糖果

【题目描述】

幼儿园里有N个小朋友，老师现在想要给这些小朋友们分配糖果，要求每个小朋友都要分到糖果。

但是小朋友们也有嫉妒心，总是会提出一些要求，比如小明不希望小红分到的糖果比他的多，于是在分配糖果的时候，老师需要满足小朋友们的K个要求。

幼儿园的糖果总是有限的，老师想知道他至少需要准备多少个糖果，才能使得每个小朋友都能够分到糖果，并且满足小朋友们所有的要求。

【输入格式】

输入的第一行是两个整数N,K。

接下来K行，表示分配糖果时需要满足的关系，每行3个数字X,A,B。

- 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。

小朋友编号从1到N。

【输出格式】

输出一行，表示老师至少需要准备的糖果数，如果不能满足小朋友们的所有要求，就输出-1。

【数据范围】

$1 \le N < 10^5$

$1 \le K \le 10^5$

$1 \le X \le 5$

$1 \le A,B \le N$

【输入样例】

```
5 7
1 1 2
2 3 2
4 4 1
3 4 5
5 4 5
2 3 5
4 5 1
```

【输出样例】

```
11
```

【分析】

本题需要求最少需要的糖果总数，因此可以通过求最长路的方法求出不等式组中各个变量的最小值。

首先将五种关系的不等式表示出来：

- $A = B \rightarrow A \geq B, B \geq A$
- $A < B \rightarrow B \geq A + 1$
- $A \geq B$
- $A > B \rightarrow A \geq B + 1$
- $A \leq B \rightarrow B \geq A$

因为每个小朋友最少需要分到一颗糖，因此附加条件为$X_i \geq 1$。

设超级源点$X_0 = 0$，则上式可以写为$X_i \geq X_0 + 1$，即从超级源点向所有点连一条长度为1的边即可。

此外，本题如果使用队列会超时，需要使用栈。

【代码】

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <stack>
using namespace std;

typedef long long LL;
const int N = 100010, M = 300010;
```

```cpp
int e[M], ne[M], d[M], h[N], idx;
int dis[N], cnt[N];
bool st[N];
int n, m;

void add(int u, int v, int w)
{
    e[idx] = v, d[idx] = w, ne[idx] = h[u], h[u] = idx++;
}

//返回值表示是否存在正环
bool spfa()
{
    memset(dis, 0x8f, sizeof dis);//求最长路,因此初始化为负无穷
    dis[0] = 0;
    stack<int> Q;//有时找环时使用栈代替队列速度更快
    Q.push(0);
    st[0] = true;
    while (Q.size())
    {
        int t = Q.top();
        Q.pop();
        st[t] = false;
        for (int i = h[t]; ~i; i = ne[i])
            if (dis[t] + d[i] > dis[e[i]])
            {
                dis[e[i]] = dis[t] + d[i];
                cnt[e[i]] = cnt[t] + 1;
                if (cnt[e[i]] >= n + 1) return true;
                if (!st[e[i]]) Q.push(e[i]), st[e[i]] = true;
            }
    }
    return false;
}

int main()
{
    cin >> n >> m;
    memset(h, -1, sizeof h);
    while (m--)
    {
        int x, a, b;

        cin >> x >> a >> b;
```

```
        if (x == 1) add(a, b, 0), add(b, a, 0);
        else if (x == 2) add(a, b, 1);
        else if (x == 3) add(b, a, 0);
        else if (x == 4) add(b, a, 1);
        else add(a, b, 0);
    }
    for (int i = 1; i <= n; i++) add(0, i, 1);
    if (spfa()) cout << -1 << endl;
    else
    {
        LL res = 0;
        for (int i = 1; i <= n; i++) res += dis[i];
        cout << res << endl;
    }
    return 0;
}

/*
X = 1: A >= B && B >= A
X = 2: B >= A + 1
X = 3: A >= B
X = 4: A >= B + 1
X = 5: B >= A
附加条件: dis[1 ~ n] >= 1, 设虚拟源点dis[0] = 0, 则dis[1 ~ n] >= dis[0] + 1
*/
```

二、AcWing 362. 区间

【题目描述】

给定n个区间$[a_i, b_i]$和n个整数c_i。

你需要构造一个整数集合Z，使得$\forall i \in [1, n]$，Z中满足$a_i \le x \le b_i$的整数x不少于c_i个。

求这样的整数集合Z最少包含多少个数。

【输入格式】

第一行包含整数n。

接下来n行，每行包含三个整数a_i, b_i, c_i。

【输出格式】

输出一个整数表示结果。

【数据范围】

$1 \le n \le 50000$

$0 \le a_i, b_i \le 50000$

$1 \le c_i \le b_i - a_i + 1$

【输入样例】

```
5
3 7 3
8 10 3
6 8 1
1 3 1
10 11 1
```

【输出样例】

```
6
```

【分析】

这一题是一定有解的，因为最坏的情况下我们可以把$1 \sim 50000$中的数据全部选上，因此不存在环。

这里可以使用前缀和的思想求解，因为前缀和中$S[0] = 0$，所以这里将a_i, b_i所在的区间范围加上一个1，区间范围变成了$[1, 50001]$，这样并不影响最终的结果。

其中，$S[i]$表示：$1 \sim i$中被选出数的个数。我们最终要求解的就是S_{50001}的最小值，因此需要使用最长路来求解。

对于S，需要满足如下条件：

1. $S_i \ge S_{i-1}, 1 \le i \le 50001$，即从区间$[1, i]$选出的数一定不会少于$[1, i-1]$选出的数；
2. $S_i - S_{i-1} \le 1 \iff S_{i-1} \ge S_i - 1$，即第$i$个数可以选和不选，个数为$1$或$0$；
3. 区间$[a, b]$中至少有c个数：$S_b - S_{a-1} \ge c \iff S_b \ge S_{a-1} + c$。

需要验证一下：从源点出发，是否一定可以走到所有的边。根据条件1，从$i-1$可以走到i，因此从0可以走到1，从1可以走到$2 \ldots$，因此0号点即为超级源点。

【代码】

```
#include <iostream>
```

```cpp
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;

const int N = 50010, M = 3 * N;
int e[M], ne[M], d[M], h[N], idx;
int dis[N];
bool st[N];
int n;

void add(int u, int v, int w)
{
    e[idx] = v, d[idx] = w, ne[idx] = h[u], h[u] = idx++;
}

void spfa()
{
    memset(dis, 0x8f, sizeof dis);
    queue<int> Q;
    Q.push(0), dis[0] = 0, st[0] = true;
    while (Q.size())
    {
        int t = Q.front();
        Q.pop();
        st[t] = false;
        for (int i = h[t]; ~i; i = ne[i])
            if (dis[t] + d[i] > dis[e[i]])
            {
                dis[e[i]] = dis[t] + d[i];
                if (!st[e[i]]) Q.push(e[i]), st[e[i]] = true;
            }
    }
}

int main()
{
    memset(h, -1, sizeof h);
    for (int i = 1; i <= 50001; i++) add(i - 1, i, 0), add(i, i - 1, -1);
    cin >> n;
    while (n--)
    {

        int a, b, c;
```

```
        cin >> a >> b >> c;
        a++, b++;//向右偏移一个单位,0号点为超级源点
        add(a - 1, b, c);
    }
    spfa();
    cout << dis[50001] << endl;
    return 0;
}

/*
利用前缀和的思想,S[i]表示区间[1 ~ i]中被选出的数的个数,S[0] = 0
将输入的a和b都加一,则需要求S[50001]的最小值
(1)S[i] >= S[i - 1],即[1 ~ i]选出的数一定不会少于[1 ~ i - 1]选出的数
(2)S[i] - S[i - 1] <= 1 -> S[i - 1] >= S[i] - 1,即第i个数可以选和不选
(3)S[b] - S[a - 1] >= c -> S[b] >= S[a - 1] + c
*/
```

三、AcWing 1170. 排队布局

【题目描述】

当排队等候喂食时，奶牛喜欢和它们的朋友站得靠近些。

农夫约翰有N头奶牛，编号从1到N，沿一条直线站着等候喂食。

奶牛排在队伍中的顺序和它们的编号是相同的。

因为奶牛相当苗条，所以可能有两头或者更多奶牛站在同一位置上。

如果我们想象奶牛是站在一条数轴上的话，允许有两头或更多奶牛拥有相同的横坐标。

一些奶牛相互间存有好感，它们希望两者之间的距离不超过一个给定的数L。

另一方面，一些奶牛相互间非常反感，它们希望两者间的距离不小于一个给定的数D。

给出M_L条关于两头奶牛间有好感的描述，再给出M_D条关于两头奶牛间存有反感的描述。

你的工作是：如果不存在满足要求的方案，输出-1；如果1号奶牛和N号奶牛间的距离可以任意大，输出-2；否则，计算出在满足所有要求的情况下，1号奶牛和N号奶牛间可能的最大距离。

【输入格式】

第一行包含三个整数N, M_L, M_D。

接下来M_L行，每行包含三个正整数A, B, L，表示奶牛A和奶牛B至多相隔L的距离。

再接下来M_D行，每行包含三个正整数A, B, D，表示奶牛A和奶牛B至少相隔D的距离。

【输出格式】

输出一个整数，如果不存在满足要求的方案，输出-1；如果1号奶牛和N号奶牛间的距离可以任意大，输出-2；

否则，输出在满足所有要求的情况下，1号奶牛和N号奶牛间可能的最大距离。

【数据范围】

$2 \leq N \leq 1000$

$1 \leq M_L, M_D \leq 10^4$

$1 \leq L, D \leq 10^6$

【输入样例】

```
4 2 1
1 3 10
2 4 20
2 3 3
```

【输出样例】

```
27
```

【分析】

根据牛在队伍中的顺序和它们的编号是相同的以及可能有两头或者更多奶牛站在同一位置上这两个条件可得：$x_i \leq x_{i+1} (1 \leq i \leq n)$，即$i+1$向$i$连一条长度为$0$的边。

假设b的编号大于a（$b > a$），若两者间的距离不超过L，则有：$x_b - x_a \leq L \iff x_b \leq x_a + L$，即$a$向$b$连一条长度为$L$的边；若两者间的距离不小于$D$，则有：$x_b - x_a \geq D \iff x_a \leq x_b - D$，即$b$向$a$连一条长度为$-D$的边。

根据第一个不等式可知以n为起点一定能走到所有的点，因此满足差分约束的条件。因此以n为起点求一遍最短路，若存在负环则不存在满足要求的方案；1号奶牛和N号奶牛间的距离可以任意大等价于将1的距离设置为0，然后再以1号点为起点求一遍最短路，最后有$dis[N] == +\infty$，说明N号点离1号点的距离可以无穷大。因此若$dis[N] != +\infty$，则$dis[N]$即为在满足所有要求的情况下，1号奶牛和N号奶牛间可能的最大距离。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;

const int N = 1010, M = 21010;
int e[M], ne[M], d[M], h[N], idx;
int dis[N], cnt[N];
bool st[N];
int n, m1, m2;

void add(int u, int v, int w)
{
    e[idx] = v, d[idx] = w, ne[idx] = h[u], h[u] = idx++;
}

bool spfa(int s)
{
    memset(dis, 0x3f, sizeof dis);
    memset(cnt, 0, sizeof cnt);
    memset(st, false, sizeof st);
    queue<int> Q;
    Q.push(s), dis[s] = 0, st[s] = true;
    while (Q.size())
    {
        int t = Q.front();
        Q.pop();
        st[t] = false;
        for (int i = h[t]; ~i; i = ne[i])
            if (dis[t] + d[i] < dis[e[i]])
            {
                dis[e[i]] = dis[t] + d[i];
                cnt[e[i]] = cnt[t] + 1;
                if (cnt[e[i]] >= n) return true;
                if (!st[e[i]]) Q.push(e[i]), st[e[i]] = true;
            }
    }
    return false;
}

int main()

{
```

```
    cin >> n >> m1 >> m2;
    memset(h, -1, sizeof h);
    for (int i = 1; i < n; i++) add(i + 1, i, 0);
    int a, b, c;
    while (m1--)//b - a <= c
    {
        cin >> a >> b >> c;
        if (b < a) swap(a, b);
        add(a, b, c);
    }
    while (m2--)//b - a >= c
    {
        cin >> a >> b >> c;
        if (b < a) swap(a, b);
        add(b, a, -c);
    }
    if (spfa(n)) cout << -1 << endl;//n号点可以走到所有点,满足差分约束条件
    else
    {
        spfa(1);//从1号点开始走,将1号点的相对位置设为0,判断到n号点距离是否为无穷大
        if (dis[n] == 0x3f3f3f3f) cout << -2 << endl;
        else cout << dis[n] << endl;
    }
    return 0;
}
```

四、AcWing 393. 雇佣收银员

【题目描述】

一家超市要每天24小时营业，为了满足营业需求，需要雇佣一大批收银员。

已知不同时间段需要的收银员数量不同，为了能够雇佣尽可能少的人员，从而减少成本，这家超市的经理请你来帮忙出谋划策。

经理为你提供了一个各个时间段收银员最小需求数量的清单$R(0), R(1), R(2), \ldots, R(23)$。

$R(0)$表示午夜$00:00$到凌晨$01:00$的最小需求数量，$R(1)$表示凌晨$01:00$到凌晨$02:00$的最小需求数量，以此类推。

一共有N个合格的申请人申请岗位，第i个申请人可以从t_i时刻开始连续工作8小时。

收银员之间不存在替换，一定会完整地工作8小时，收银台的数量一定足够。

现在给定你收银员的需求清单，请你计算最少需要雇佣多少名收银员。

【输入格式】

第一行包含一个不超过20的整数，表示测试数据的组数。

对于每组测试数据，第一行包含24个整数，分别表示$R(0), R(1), R(2), \ldots, R(23)$。

第二行包含整数N。

接下来N行，每行包含一个整数t_i。

【输入格式】

每组数据输出一个结果，每个结果占一行。

如果没有满足需求的安排，输出`No Solution`。

【数据范围】

$0 \leq R(0) \leq 1000$

$0 \leq N \leq 1000$

$0 \leq t_i \leq 23$

【输入样例】

```
1
1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
5
0
23
22
1
10
```

【输出样例】

```
1
```

【分析】

由于牵扯到$[i-8+1, i]$这段区间的和的约束，所以用前缀和更好表达一些。

设$num[i]$表示i时刻有多少人申请上岗，$x[i]$为i时刻实际上岗的人数，s为x的前缀和数组。

则应该满足的约束条件是：

1. 上岗人数不能负数，即$s[i]-s[i-1]\geq 0 \iff s[i]\geq s[i-1]$，即$add(i-1,i,0)$
2. 实际上岗人数不能超过申请人数，即$s[i]-s[i-1]\leq num[i] \iff s[i-1]\geq s[i]-num[i]$，即$add(i,i-1,-num[i])$
3. i时刻所在人数，即$[i-7,i]$区间内的上岗总人数要大于等于最小需求$R[i]$，由于存在环，即23到24，再到0时刻，所以要分类讨论：

   （1）当$i\geq 8$时，$s[i]-s[i-8]\geq R[i] \iff s[i]\geq s[i-8]+R[i]$，即$add(i-8,i,R[i])$

   （2）当$i\leq 7$时，$s[i]+s[24]-s[i+16]\geq R[i] \iff s[i]\geq s[i+16]+R[i]-s[24]$，不会连边了hhh

最后一种约束关系我们不会连边的原因无非是出现了三个变量，但我们可以发现：

所有最后一种约束关系都有$s[24]$变量，其实这个东西就是我们求的答案，所以我们可以枚举$s[24]$的值，把它变成常量就行啦！然后就可以$add(16+i,i,R[i]-s[24])$。

***Tips*：**

根据第一个不等式可以发现0肯定能走到所有点，所以不用创造超级源点了，只需从0点出发跑最长路即可。

不要忘了$s[24]$等于我们枚举的数s（要严格等于，实现是**大于等于+小于等于**）：

$s[24]\leq s$，即$add(24,0,-s)$

$s[24]\geq s$，即$add(0,24,s)$

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;

const int N = 30, M = 100;
int e[M], ne[M], d[M], h[N], idx;
int dis[N], cnt[N];
int num[N], R[N];
bool st[N];
int n;

void add(int u, int v, int w)
{
```

```
    e[idx] = v, d[idx] = w, ne[idx] = h[u], h[u] = idx++;
}

void build(int s)//s表示S24
{
    memset(h, -1, sizeof h);
    idx = 0;
    for (int i = 1; i <= 24; i++) add(i - 1, i, 0), add(i, i - 1, -
num[i]);
    for (int i = 8; i <= 24; i++) add(i - 8, i, R[i]);
    for (int i = 1; i <= 7; i++) add(i + 16, i, -s + R[i]);
    //S24 == s -> S24 >= s && S24 <= s
    add(0, 24, s), add(24, 0, -s);
}

bool spfa(int s)
{
    build(s);
    memset(dis, 0x8f, sizeof dis);
    memset(cnt, 0, sizeof cnt);
    memset(st, false, sizeof st);
    queue<int> Q;
    Q.push(0), dis[0] = 0, st[0] = true;//0号点可以走到所有点
    while (Q.size())
    {
        int t = Q.front();
        Q.pop();
        st[t] = false;
        for (int i = h[t]; ~i; i = ne[i])
            if (dis[t] + d[i] > dis[e[i]])
            {
                dis[e[i]] = dis[t] + d[i];
                cnt[e[i]] = cnt[t] + 1;
                if (cnt[e[i]] >= N) return true;
                if (!st[e[i]]) Q.push(e[i]), st[e[i]] = true;
            }
    }
    return false;
}

int main()
{

    int T;
```

```
    cin >> T;
    while (T--)
    {
        memset(num, 0, sizeof num);
        for (int i = 1; i <= 24; i++) cin >> R[i];
        cin >> n;
        while (n--) { int t; cin >> t; num[t + 1]++; }
        bool flag = false;//表示是否有解
        for (int i = 0; i <= 1000; i++)//枚举S24的取值
            if (!spfa(i)) { flag = true; cout << i << endl; break; }
        if (!flag) puts("No Solution");
    }
    return 0;
}
```