一、简答

1. n 个变量， m 个不等式约束通常转化为 n 个顶点， m 条边的有向图，再通过 spfa 求解最短/长路，转化通常是 $O(m)$, spfa 复杂度为 $O(k*m)$ ，其中 k 通常为一个较小常数。因此总复杂度为 $O(km)$.

二、作业H8

C. 种酸奶

题目大意

求满足一定区间约束的01串中1的最多个数。

解法

题为求约束系统的最大值问题。最大值问题可以转化为最短路：对于节点 i ,我们要找到 i 的最小值，也就是其最大上界，对应的是 $x_i - x_s <= dis[i]$ 中 dis[i] 为两点最短路。
对于不等式 $x_i - x_j <= c$ ，对应最短路中的意义为从 j 开始松弛其邻接的边（长度为 c ， 连接 i 点），判据为 $x_i - x_j > c$ ，因此我们从点 j 向 i 连接一条长度为 c 的边。

时间复杂度

$O(n+km)$

代码

```cpp
#include<iostream>
#include<vector>
#include<queue>
using namespace std;
using ll = long long;

// a = a - 1
// k == 1: cb - ca <= c
// -> cb <= ca + c
// k == 2: cb - ca >= c
// k == 3: cb - ca <= c - 1
// k == 4: cb - ca >= c + 1
// max(cn)
// 0 <= dis[i] - d[i - 1] <= 1

const int N = 1010;
int n, m;
int dis[N], cnt[N];
vector<pair<int, int> > g[N];
bool vis[N];

void spfa(int s) {
    for (int i = 1; i <= n; i++) {
        vis[i] = false;
        dis[i] = INT32_MAX;
        cnt[i] = 0;
    }

    dis[s] = 0;
    vis[s] = true;
    queue<int> q;
    q.push(s);

    while (!q.empty()) {
        int u = q.front();
        q.pop();
        vis[u] = false;

        for (auto i : g[u]) {
            int v = i.first, l = i.second;
            if (dis[v] > dis[u] + l) {
                dis[v] = dis[u] + l;
                cnt[v] = cnt[u] + 1;

                if (cnt[v] >= n) {
                    cout << "impossible\n";
                    return;
                }

                if (!vis[v]) {
```

```
                    vis[v] = true;
                    q.push(v);
                }
            }
        }
    }

    if (dis[n] == INT32_MAX) {
        cout << "impossible\n";
    }
    else cout << dis[n] << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> n >> m;

    int k, a, b, c;
    for (int i = 1; i <= m; i++) {
        cin >> k >> a >> b >> c;
        a--;
        switch (k) {
            case 1: g[a].push_back({b, c});
                    break;
            case 2: g[b].push_back({a, -c});
                    break;
            case 3: g[a].push_back({b, c - 1});
                    break;
            case 4: g[b].push_back({a, -c - 1});
                    break;
            case 5: g[a].push_back({b, c});
                    g[b].push_back({a, -c});
                    break;
        }
    }

    for (int i = 1; i <= n; i++) {
        g[i - 1].push_back({i, 1});
        g[i].push_back({i - 1, 0});
    }

    spfa(0);
    return 0;
}
```

C. 种酸奶

题目大意

有 n 个人，每个人都有一个编号，从 1 到 n 。
如果 A 得知一个消息，那么他一定会告诉 B 。
问最少把消息告诉几个人，能让所有人得知这个消息。

解法

强连通缩点，求入度为 0 的点的数量。

时间复杂度

$O(n+m)$

代码

```cpp
#include<iostream>
#include<vector>
#include<algorithm>
#define N 1000010
using namespace std;
using ll = long long;

// SCC 缩点

vector<int> g1[N], g2[N], g[N];
bool vis[N];
int dfn[N], scc[N], sz[N];
int cnt, tot;
int d[N];

void dfs(int u) {
    vis[u] = 1;
    for (int v : g1[u]) {
        if (!vis[v]) dfs(v);
    }
    dfn[++cnt] = u;
}

void dfs1(int u, int c) {
    scc[u] = c;
    for (int v : g2[u]) {
        if (!scc[v]) dfs1(v, c);
    }
}

void solve() {
    int n, m;
    cin >> n >> m;

    int a, b;
    for (int i = 1; i <= m; i++) {
        cin >> a >> b;
        g1[a].push_back(b);
        g2[b].push_back(a);
    }

    for (int i = 1; i <= n; i++) {
        if (!vis[i]) dfs(i);
    }

    for (int i = n; i > 0; i--) {
        if (!scc[dfn[i]]) dfs1(dfn[i], ++tot);
    }

    for (int i = 1; i <= n; i++) {
```

```
        sz[scc[i]]++;
        for (int j : g1[i]) {   // 对原图进行缩点
            if (scc[j] != scc[i]) {
                g[scc[i]].push_back(scc[j]);
            }
        }
    }

    for (int i = 1; i <= tot; i++) {
        sort(g[i].begin(), g[i].end());
        vector<int>::iterator newEnd;
        newEnd = unique(g[i].begin(), g[i].end());
        g[i].erase(newEnd, g[i].end());
        for (int v : g[i]) d[v]++;
    }

    int ans = 0;
    for (int i = 1; i <= tot; i++) {
        if (!d[i]) {
            ans++;
        }
    }
    cout << ans << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    solve();
    return 0;
}
```