一、简答

1. 有可能，当两点不连通时它们的距离无穷大，我们一般用一个较大值 INF 表示。观察 $f[k][x][y]=min(f[k-1][x][y],f[k-1][x][k]+f[k-1][k][y])$ 我们发现当更新 $f[k][*][*]$ 时，始终满足关系 $f[k-1][x][k]==f[k][x][k]$ 和 $f[k-1][k][y]==f[k][k][y]$ ， 因此可以表达成 $f[x][y]=min(f[x][y],f[x][k]+f[k][y])$.
2. 当最短路的长度或者一些边被松弛次数超过 $n-1$,又或者在第 n 次松弛操作时还存在边能够被成功松弛。

二、作业H7

B. 买菜

题目大意

求两个线段集合的相交长度。

解法

方法一： 可以直接双指针扫描两个集合进行判断。
方法二： 珂朵莉树维护线段集合，将小 H 经过的区间赋值为1,将小 W 经过的区间进行求和。

时间复杂度

方法一： $O(2*n)$
方法二： $O(nlogn)$

代码

```cpp
// 方法一:
vector<pair<int, int> > a(2005);

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

    int n;
    cin >> n;

    int x, y;
    for (int i = 0; i < n; i++) {
        cin >> x >> y;
        a[i] = {x, y};
    }

    int ans = 0, j = 0;
    for (int i = 0; i < n; i++) {
        cin >> x >> y;
        if (a[j].first >= y) {
            continue;
        }

        TAG: while (j < n && a[j].second <= x) {
            j++;
        }

        if (j < n) ans += max(min(a[j].second, y) - max(a[j].first, x), 0);

        if (j < n && y > a[j].second) {
            j++;
            goto TAG;
        }

        // cout << "(x, y): " << x << ' ' << y << '\n';
        // cout << "(fi, se):" << a[j].first << ' ' << a[j].second << '\n';
    }
    cout << ans;
    return 0;
}

// 方法二:
struct node {
    ll l ,r;
    mutable ll v;
    node(ll l, ll r, ll v) : l(l), r(r), v(v) {}
    bool operator<(const node &a) const { return l < a.l; }
};

set<node> tree;
```

```cpp
set<node>::iterator split(ll pos) {
    auto it = tree.lower_bound(node(pos, 0, 0));
    if (it != tree.end() && it->l == pos)
        return it;
    it--;
    ll L = it->l, R = it->r, V = it->v;
    tree.erase(it);
    tree.insert(node(L, pos - 1, V));
    return tree.insert(node(pos, R, V)).first;
}

void assign(ll l, ll r, ll v) {
    auto end = split(r + 1), begin = split(l);
    tree.erase(begin, end);
    tree.insert(node(l, r, v));
}

void solve() {
    int n; cin >> n;
    tree.insert(node(1, n, 0));
    int a, b;
    for (int i = 1; i <= n; i++) {
        cin >> a >> b;
        assign(a, b, 1);
    }

    int ans = 0;
    for (int i = 1; i <= n; i++) {
        cin >> a >> b;
        auto end = split(b + 1), begin = split(a);
        for (auto j = begin; j != end; j++) {
            if (j->v == 1)  {
                ans += (j->r - j->l);
                // cout << j->l << ' ' << j->r << '\n';
            }
        }
    }
    cout << ans;
}

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

C. 穿越虫洞

题目大意

判断图中是否存在负环。

解法

spfa,判断是否存在最短路长度 $>=n$.

时间复杂度

$O(km)$

代码

```cpp
vector<pair<int, int> > g[505];
int cnt[505], dis[505];
bool vis[505];
int n, m, w;

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

    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]) {
            if (dis[i.second] > dis[u] + i.first) {
                dis[i.second] = dis[u] + i.first;
                cnt[i.second] = cnt[u] + 1;
                if (cnt[i.second] >= n) {
                    cout << "YES\n";
                    return;
                }
                if (!vis[i.second]) {
                    vis[i.second] = 1;
                    Q.push(i.second);
                }
            }
        }
    }

    cout << "NO\n";
}

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

    int F; cin >> F;
    while (F--) {

        cin >> n >> m >> w;
```

```
        for (int i = 1; i <= n; i++) {
            g[i].clear();
        }

        int s, e, t;
        for (int i = 1; i <= m; i++) {
            cin >> s >> e >> t;
            g[s].push_back({t, e});
            g[e].push_back({t, s});
        }

        for (int i = 1; i <= w; i++) {
            cin >> s >> e >> t;
            g[s].push_back({-t, e});
        }

        spfa(1);
    }

    return 0;
}
```