

# Bellman-ford + SPFA

## 人员

赵熙羽、司云心、于子珈、陈洛冉、杨咏丞、杨瑾硕、董浩桢、牟茗、秦显森、牛同泽 到课, 周子一 线上

## 上周作业检查

上周作业链接: <https://cppoj.kids123code.com/contest/1910>

#	用户名	姓名	编程分	时间	A	B	C	D
1	yuzijia1	于子珈	400	4772	100	100	100	100
2	chenluoran	陈洛冉	400	10586	100	100	100	100
3	muming	牟茗	400	11268	100	100	100	100
4	zhaoxiyu	赵熙羽	400	11846	100	100	100	100
5	yangyongcheng	杨咏丞	306	1146	100	6	100	100
6	zhousiyi	周子一	300	4963	100	100		100
7	qinxiansen	秦显森	225	6421	100	100	25	
8	niutongze	牛同泽	200	217	100			100
9	yangjinshuo	杨瑾硕	200	6704	100	100		
10	sultianyi	隋天乙	200	7160	100	100		
11	xiekai	谢亚锴	110	44	100	10		
12	siyunxin	司云心	110	45	100	您正在共享屏幕	结束共享	
13	donghaozhen	董浩桢	100	45	100			

## 本周作业

<https://cppoj.kids123code.com/contest/2020> (课上讲了 A ~ B 题, 课后作业是 C 题)

## 课堂表现

今天讲了 Bellman-ford 和 SPFA 两个算法, 同学们上课听讲都很认真, 吸收的也都很不错。

## 课堂内容

### 拆地毯 (上周作业)

最大生成树做法, 只选 K 条边即可

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 2e5 + 5;
int f[maxn];
struct node {
    int u, v, value;
    bool operator < (const node& p) const { return value < p.value; }
}
```

```

};

int fFind(int x) {
    if (f[x] != x) f[x] = fFind(f[x]);
    return f[x];
}

void kruskal(vector<node>& vec, int n, int K) {
    sort(vec.begin(), vec.end()); reverse(vec.begin(), vec.end());
    for (int i = 1; i <= n; ++i) f[i] = i;

    int sum = 0, cnt = 0;
    for (node it : vec) {
        int u = it.u, v = it.v, value = it.value;
        int pu = fFind(u), pv = fFind(v);
        if (pu == pv) continue;
        f[pu] = pv; sum += value; ++cnt;
        if (cnt == K) break;
    }

    cout << sum << endl;
}

int main()
{
    int n, m, K; cin >> n >> m >> K;
    vector<node> vec;
    while (m -- ) {
        int u, v, value; cin >> u >> v >> value;
        vec.push_back({u, v, value});
    }
    kruskal(vec, n, K);
    return 0;
}

```

## 【模板】单源最短路径（弱化版）

bellman-ford 模板, 会超时, 70 分代码

```

#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 1e4 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct Edge {
    int from, to, value;
};
int dis[maxn];

```

```

signed main()
{
    int n, m, id; cin >> n >> m >> id;
    vector<Edge> edges;
    while (m -- ) {
        int a, b, c; cin >> a >> b >> c; edges.push_back({a,b,c});
    }

    memset(dis, 0x3f, sizeof(dis)); dis[id] = 0;
    for (int i = 1; i <= n-1; ++i) {
        for (Edge it : edges) dis[it.to] = min(dis[it.to], dis[it.from]+it.value);
    }

    int limit = (1LL<<31) - 1;
    for (int i = 1; i <= n; ++i) {
        if (dis[i] == inf) cout << limit << " ";
        else cout << dis[i] << " ";
    }
    cout << endl;
    return 0;
}

```

## SPFA 模板

```

#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 1e4 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
const int limit = (1LL<<31) - 1;
struct node {
    int to, value;
};
vector<node> vec[maxn];
int dis[maxn];
bool st[maxn];

signed main()
{
    int n, m, id; cin >> n >> m >> id;
    while (m -- ) {
        int a, b, c; cin >> a >> b >> c; vec[a].push_back({b,c});
    }

    queue<int> q;
    memset(dis, 0x3f, sizeof(dis));
    q.push(id); dis[id] = 0; st[id] = true;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        st[u] = false;

```

```

    for (node it : vec[u]) {
        if (dis[it.to] > dis[u]+it.value) {
            dis[it.to] = dis[u]+it.value;
            if (!st[it.to]) q.push(it.to), st[it.to] = true;
        }
    }
}

for (int i = 1; i <= n; ++i) {
    if (dis[i] == inf) cout << limit << " ";
    else cout << dis[i] << " ";
}
cout << endl;
return 0;
}

```

## 【模板】负环

### Bellman-ford 判负环

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 2e3 + 5;
const int inf = 0x3f3f3f3f;
struct Edge {
    int from, to, value;
};
int dis[maxn], f[maxn];
bool st[maxn];

void solve() {
    vector<Edge> edges;
    int n, m; cin >> n >> m;
    while (m -- ) {
        int a, b, c; cin >> a >> b >> c;
        edges.push_back({a,b,c});
        if (c >= 0) edges.push_back({b,a,c});
    }

    memset(dis, 0x3f, sizeof(dis)); dis[1] = 0;
    memset(st, false, sizeof(st)); st[1] = true;
    for (int i = 1; i <= n; ++i) {
        for (Edge it : edges) {
            int from = it.from, to = it.to, value = it.value;
            dis[to] = min(dis[to], dis[from]+value);
            st[to] |= st[from];
        }
        if (i == n-1) {
    
```

```

        for (int j = 1; j <= n; ++j) f[j] = dis[j];
    }

    for (int i = 1; i <= n; ++i) {
        if (st[i] && f[i] != dis[i]) { cout << "YES" << endl; return; }
    }
    cout << "NO" << endl;
}

int main()
{
    int T; cin >> T;
    while (T -- ) solve();
    return 0;
}

```

## SPFA 判负环

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 2e3 + 5;
const int inf = 0x3f3f3f3f;
struct node {
    int to, value;
};
vector<node> vec[maxn];
int dis[maxn], cnt[maxn];
bool st[maxn];

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

    for (int i = 1; i <= n; i ++ ) vec[i].clear();
    memset(dis, 0x3f, sizeof(dis));
    memset(cnt, 0, sizeof(cnt));
    memset(st, false, sizeof(st));

    while (m -- ) {
        int a, b, c; cin >> a >> b >> c; vec[a].push_back({b,c});
        if (c >= 0) vec[b].push_back({a,c});
    }

    queue<int> q;
    q.push(1); dis[1] = 0; cnt[1] = 1; st[1] = true;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        st[u] = false;

        for (node it : vec[u]) {
            if (dis[it.to] > dis[u] + it.value) {
                dis[it.to] = dis[u] + it.value;
                if (cnt[it.to] < m) {
                    q.push(it.to);
                    cnt[it.to]++;
                }
            }
        }
    }
}

```

```
if (dis[it.to] > dis[u]+it.value) {  
    dis[it.to] = dis[u]+it.value; cnt[it.to]++;
    if (cnt[it.to] >= n) { cout << "YES" << endl; return; }
    if (!st[it.to]) q.push(it.to), st[it.to] = true;
}  
}  
}  
  
cout << "NO" << endl;  
}  
  
int main()  
{  
    int T; cin >> T;  
    while (T -- ) solve();  
    return 0;  
}
```