

prim

人员

赵熙羽、陈洛冉、谢亚锴、牛同泽、杨瑾硕、周子一、司云心、于子珈、秦显森 到课, 李子瀚 线上

上周作业检查

上周作业链接: <https://cppoj.kids123code.com/contest/2239>



The screenshot shows a competition results page with a table of scores. The table has columns for rank (#), username, name, programming score, time, and five specific problem scores (A, B, C, D, E). The top row is highlighted in blue.

#	用户名	姓名	编程分	时间	A	B	C	D	E
1	lizihan	李子瀚	500	446	100	100	100	100	100
2	niutongze	牛同泽	400	738	100	100	100	100	100
3	zhaoxiyu	赵熙羽	400	1113	100	100	100	100	100
4	yuzijia1	于子珈	300	350	100	100	100	0	
5	chenluoran	陈洛冉	300	393	100	100	100		
6	yangjinshuo	杨瑾硕	200	100	100	100			
7	xieyakai	谢亚锴	200	101	100	100	0		
8	zhouziyi	周子一	200	239	100	100			
9	suitianyi	隋天乙	100	20	100				
	siyunxin	司云心	100	20	100				

下方有一个对话框提示“说点什么...”

本周作业

<https://cppoj.kids123code.com/contest/2330> (课上讲了 A ~ D 题, 课后作业是 E 题)

课堂表现

今天的重点内容是今天的 C D 题, D 题同学们课上都没做完, 课下要把这道题目补完。

课堂内容

旅行计划 (上周作业)

拓扑排序 + 最长路

f[i]: 以 i 结尾的最长路是多少

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

using namespace std;

const int maxn = 1e5 + 5;
vector<int> vec[maxn];
int deg[maxn], f[maxn];
```

```

int main()
{
    int n, m; cin >> n >> m;
    while (m -- ) {
        int x, y; cin >> x >> y; vec[x].push_back(y); ++deg[y];
    }

    queue<int> q;
    for (int i = 1; i <= n; ++i) {
        if (!deg[i]) q.push(i), f[i] = 1;
    }
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (int i : vec[u]) {
            --deg[i]; f[i] = max(f[i], f[u]+1);
            if (!deg[i]) q.push(i);
        }
    }
}

for (int i = 1; i <= n; ++i) cout << f[i] << endl;
return 0;
}

```

【模板】最小生成树

prim, $O(n^2)$ 时间复杂度的 最小生成树 模板做法

选 n 次, 每次选最近的一个点出来

prim 算法适用于稠密图, kruskal 算法适用于稀疏图

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 5000 + 5;
const int inf = 0x3f3f3f3f;
int w[maxn][maxn];
int dis[maxn];
bool st[maxn];
int n, m;

int prim() {
    memset(dis, 0x3f, sizeof(dis));

    int res = 0;
    for (int i = 0; i < n; i++) {
        int id = -1;
        for (int j = 1; j <= n; j++) {
            if (st[j]) continue;

```

```

        if (id == -1 || dis[j] < dis[id]) id = j;
    }

    if (i && dis[id] == inf) return -1;
    if (i) res += dis[id];

    st[id] = true;
    for (int j = 1; j <= n; j++) {
        if (!st[j]) dis[j] = min(dis[j], w[id][j]);
    }
}
return res;
}

int main()
{
    cin >> n >> m;
    memset(w, 0x3f, sizeof(w));
    while (m -- ) {
        int a, b, c; cin >> a >> b >> c;
        w[a][b] = w[b][a] = min(w[a][b], c);
    }

    int res = prim();

    if (res == -1) cout << "orz" << endl;
    else cout << res << endl;
    return 0;
}

```

买礼物

建一个点 0 的虚点, 点 0 到点 i 的距离为 A, 然后用 prim 求最小生成树

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 500 + 5;
const int inf = 0x3f3f3f3f;
int w[maxn][maxn], dis[maxn];
bool st[maxn];
int A, n;

int prim() {
    memset(dis, 0x3f, sizeof(dis));

    int res = 0;
    for (int i = 0; i < n+1; ++i) {
        int id = -1;
        for (int j = 0; j <= n; ++j) {

```

```

        if (st[j]) continue;
        if (id == -1 || dis[j] < dis[id]) id = j;
    }

    if (i && dis[id] == inf) return -1;
    if (i) res += dis[id];

    st[id] = true;
    for (int j = 0; j <= n; ++j) {
        if (!st[j]) dis[j] = min(dis[j], w[id][j]);
    }
}
return res;
}

int main()
{
    cin >> A >> n;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            cin >> w[i][j];
            if (!w[i][j]) w[i][j] = A;
        }
    }
    for (int i = 1; i <= n; ++i) w[0][i] = w[i][0] = A;

    cout << prim() << endl;
    return 0;
}

```

最短路计数

在 bfs 维护 dis 最短路的基础上, 额外维护一个 f 数组

其中, f[i] 代表到 i 最短路的条数

如果 $u \rightarrow i$ 是第一次更新到 i 的最短路, 那么此时 $f[i] = f[u]$

否则如果 $u \rightarrow i$ 等于 i 的最短路, 那么此时 $f[i] += f[u]$

```

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

using namespace std;

const int maxn = 1e6 + 5;
const int mod = 100003;
const int inf = 0x3f3f3f3f3f3f3f3f;
int dis[maxn], f[maxn];
vector<int> vec[maxn];

signed main()

```

```

{
    ios::sync_with_stdio(false);
    cin.tie(0);

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

    memset(dis, -1, sizeof(dis));
    queue<int> q;
    q.push(1); dis[1] = 0; f[1] = 1;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (int i : vec[u]) {
            if (dis[i] == -1) {
                q.push(i); dis[i] = dis[u]+1; f[i] = f[u];
            } else if (dis[i] == dis[u]+1) f[i] = (f[i] + f[u]) % mod;
        }
    }

    for (int i = 1; i <= n; ++i) cout << f[i] << "\n";
    return 0;
}

```

路径统计

此题有重边, 需要先对边进行去重操作

解题方法跟上道题一模一样, 就是把用 bfs 求最短路的过程, 替换成用 dijkstra 求最短路

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 2000 + 5;
const int inf = 0x3f3f3f3f;
struct eInfo {
    int to, value;
};
vector<eInfo> vec[maxn];
int w[maxn][maxn];

struct node {
    int id, d;
    bool operator < (const node& p) const { return d < p.d; }
    bool operator > (const node& p) const { return d > p.d; }
};
int dis[maxn];
bool st[maxn];

```

```
int f[maxn];

void dijkstra(int _st) {
    memset(dis, 0x3f, sizeof(dis)), memset(st, false, sizeof(st));
    priority_queue<node, vector<node>, greater<node>> q;
    q.push({_st, 0}); dis[_st] = 0; f[_st] = 1;
    while (!q.empty()) {
        node u = q.top(); q.pop();
        int id = u.id, d = u.d;
        if (st[id]) continue;
        st[id] = true;

        for (eInfo it : vec[id]) {
            if (dis[it.to] > d+it.value) {
                dis[it.to] = d+it.value; q.push({it.to, dis[it.to]}); f[it.to] = f[id];
            } else if (dis[it.to] == d+it.value) f[it.to] += f[id];
        }
    }
}

int main()
{
    int n, m; cin >> n >> m;
    memset(w, 0x3f, sizeof(w));
    while (m -- ) {
        int a, b, c; cin >> a >> b >> c; w[a][b] = min(w[a][b], c);
    }

    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
            if (w[i][j] != inf) vec[i].push_back({j,w[i][j]});
        }
    }

    dijkstra(1);
    if (dis[n] == inf) cout << "No answer" << endl;
    else cout << dis[n] << " " << f[n] << endl;
    return 0;
}
```