1. **邻接矩阵存图：**

//邻接矩阵存图

#include<iostream>

#include<algorithm>

#include<cstdio>

using namespace std;

#define INF 0x3f3f3f3f

const int maxn = 1e3 + 100;

int n, m;

int graph[maxn][maxn];

int main() {

while (~scanf("%d%d", &n, &m)) {

if (n == 0 && m == 0) return 0;

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

for (int j = 1; j <= n; j++) {

graph[i][j] = INF; //当为有权图时赋值为：INF，当为无权图时赋值为：0

}

}

while (m--) {

int a, b, c;

scanf("%d%d%d", &a, &b, &c);

graph[a][b] = c;

graph[b][a] = c; //当为有向图时，仅写第一个(即：graph[a][b] = c)

}

}

return 0;

}

1. **邻接表存图：**

//邻接表存图

#include<iostream>

#include<algorithm>

#include<cstdio>

#include<vector>

using namespace std;

const int maxn = 1e5 + 100;

int n, m;

//定义边

struct edge {

int from, to, w; //边：起点 from，终点 to，权值 w

edge(int a, int b, int c) {

from = a; to = b; w = c; //对边进行赋值

}

};

vector<edge>e[maxn];

void init() {

for (int i = 1; i <= n; i++) e[i].clear(); //初始化

}

int main() {

while (~scanf("%d%d", &n, &m)) {

if (n == 0 && m == 0) return 0;

init();

while (m--) {

int a, b, c;

scanf("%d%d%d", &a, &b, &c);

//对无向图的边的赋值、存储（如下两行）：

e[a].push\_back(edge(a, b, c));

e[b].push\_back(edge(b, a, c));

//对有向图的边的赋值、存储（如下一行）：

e[a].push\_back(edge(a, b, c));

}

}

return 0;

}

1. **链式向前星**

//链式向前星存图

#include<iostream>

#include<algorithm>

#include<cstdio>

using namespace std;

const int maxn = 1e5 + 100;

int n, m;

//定义边

struct edge {

int to, next, w; //边：终点 to、权值 w、下一个边 next.起点放在head[]中

}edge[maxn];

int head[maxn]; //head[u[指向结点u的第一个边的存储位置

int cnt; //记录edge[]的末尾位置，新加入的边放在末尾

void init() { //初始化

for (int i = 1; i <= maxn - 1; i++) {

edge[i].next = -1; //-1：结束，没有下一个边

head[i] = -1; //-1：不存在从结点i出发的边

}

cnt = 0;

}

void addedge(int u, int v, int w) {

edge[cnt].to = v;

edge[cnt].w = w;

edge[cnt].next = head[u]; //指向结点u上一次存的边的位置

head[u] = cnt++; //更新结点u最新边的存放位置：就是edge的末尾

}

//补充：

//遍历结点i的所有邻居

//for (int i = head[u]; ~i; i = edge[i].next) {}

int main() {

while (~scanf("%d%d", &n, &m)) {

if (n == 0 && m == 0) return 0;

init();

while (m--) {

int a, b, c;

scanf("%d%d%d", &a, &b, &c);

addedge(a, b, c);

addedge(b, a, c); //当为有向图时，仅写第一个(即：addedge(a, b, c))

}

}

return 0;

}