## ACM Contest and Blackout

### Input:

N, M

N <= 100

### Output:

S1, S2

### Sample test:

### 

### Idea:

Tìm cây khung nhỏ nhất => Prim

Tìm cây khung nhỏ nhì

Lần lượt bỏ các cạnh trên cây khung nhỏ nhất => Prim tìm MST mới (nếu có)

Bỏ 1 canh trên cây khung nhỏ nhất -> Prim -> mst mới != mst cũ

### Code:

class Node {

Int node;

Int weight;

Int compare(Node u) {

return weight < u.weight;

}

}

cosnt int INF = 10^9

int dist[109], path[109], visited[109];

Listr<Node> graph[109];

function prim(int s) {

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

Visited[i] = 0;

Path[i] = -1;

Dist[i] = INF;

}

dist[s] = 0;

Priority\_queue<Node> pq;

pq.push(Node(s, 0));

while(!pq.empty()) {

Int u = pq.top().node;

visited[u] = 1;

for(int i = 0; i <= graph[u].size(); ++i) {

if(mark[u][i] == 1) {

continue;

}

Int v = graph[u][i].node;

Int w = graph[u][i].weight;

if(!visited[v] && dist[v] > weight) {

Dist[v] = weight;

Path[v] = u;

pq.push(Node(v, w));

}

}

}

Int sum = 0;

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

if(dist[i] == INF) {

Return INF;

}

Sum += dist[i];

}

Return sum;

}

Function main() {

input(numTescase);

while(numTestcase > 0) {

numTestcase--;

input(n, m);

for(int i: 1->m) {

input(u, v, w);

graph[u].push(Node(v, w));

graph[v].push(Node(u, w));

}

Int mst1 = prim(1); Nlog(M)

Int tempPath[109];

tempPath = path;

Int mst2 = INF;

// Gia su canh khac biet giu mst1 va mst 2 la canh noi v voi path[v]

for(int v = 2; v <= n; ++v) { O(N)

Int u = path[v];

Int w = dist[v];

Int idxu, idxv;

for(int i = 0; i < graph[u].size()l; ++i) {

if(graph[u][i].node == v && graph[u][i].weight == w) {

Mark[u][i] = 1;

Idxu = i;

break;

}

}

for(int i = 0; i < graph[v].size()l; ++i) {

if(graph[v][i].node == v && graph[v][i].weight == w) {

Mark[v][i] = 1;

Idxv = i;

break;

}

}

Int tempMst = prim(1); O(NlogM)

mst2 = min(mst2, tempMst);

Mark[u][idxu] = 0;

Mark[v][idxv] = 0;

}

output(mst1, mst2);

}

}

Độ phức tapj: O(T \* NlogM + N\*NlogM + M) = O(T \* N\*NlogM) (15\*10^4log10^4)