## XYZZY Analysis

### Input:

* Multi testcase
* End with -1
* N (room) <= 100
* -100 <= weight <= 100

### Output:

* winnable/hopeless
* winnable: có đường đi từ 1 -> n , energy >= 0 trên đường

### Ý tưởng:

Tìm đường đi có energy lớn nhất từ 1 đến N.

dist[u] + w >= dist[v] : cập nhập v

=> Dijkstra hoặc Bellman Fold

### Code:

Function DFS(int u) {

for(int v: adj[u]) {

if(dist[v] != INF) {

dist[v] = INF;

DFS(v);

}

}

}

function Bellman(int S) {

Int dist[];

for(i:1 -> MAXN) {

Dist[i] = -INF;

}

dist[1] = 100;

/\*

Duyet V-1 lan -> mang dist

\*/

for(int i: 1->n-1) {

for(int j: 1 -> m) {

Int u = graph[i].firstNode;

Int v = graph[i].secondNode;

Int w = weight[v];

if(dist[u] != -INF && dist[u] + w >= dist[v] && dist[u] + w >= 0) {

dist[v] = dist[u] + w;

}

}

}

/\*

Duyet m canh -> check chu trinh am, danh dau dinh thuoc chu trinh am

\*/

for(i : 1-> m) {

Int u = graph[i].firstNode;

Int v = graph[i].secondNode;

Int w = weight[v];

if(dist[u] != -INF && dist[u] + w >= dist[v] && dist[u] + w >= 0) {

dist[v] = INF;

}

}

/\*

Voi moi dinh thuoc chu trinh am, BFS/DFS tim moi dinh lien quan toi chu trinh am

\*/

for(i: 1-> n) {

if(dist[i] == INF) {

DFS(i);

}

}

}

Function main() {

while(true) {

input(N);

If ( N == -1 ) break;

for(i: 1-> N) {

Int w, numNode, idNode;

input(w);

input(numNode);

m += numNode;

for(j: 1 -> numNode) {

input(nodeId)

graph.add(Edge(i, nodeId));

adj[i].add(j):

Weight[i] = w;

}

}

Bellman(1);

If(dist[n] != -INF) {

output(“winnable”);

} else {

output(“hopeless);

}

}

}