#include <iostream>

#include <vector>

#include <stack>

#include <climits>

using namespace std;

struct Edge {

int to;

int weight;

};

void topologicalSortUtil(int v, vector<bool>& visited, stack<int>& Stack, const vector<vector<Edge>>& graph) {

visited[v] = true;

for (const Edge& edge : graph[v]) {

if (!visited[edge.to]) {

topologicalSortUtil(edge.to, visited, Stack, graph);

}

}

Stack.push(v);

}

void longestPath(int src, const vector<vector<Edge>>& graph, int V) {

stack<int> Stack;

vector<int> distances(V, INT\_MIN);

vector<bool> visited(V, false);

for (int i = 0; i < V; i++) {

if (!visited[i]) {

topologicalSortUtil(i, visited, Stack, graph);

}

}

distances[src] = 0;

while (!Stack.empty()) {

int u = Stack.top();

Stack.pop();

if (distances[u] != INT\_MIN) {

for (const Edge& edge : graph[u]) {

if (distances[edge.to] < distances[u] + edge.weight) {

distances[edge.to] = distances[u] + edge.weight;

}

}

}

}

for (int i = 0; i < V; i++) {

(distances[i] == INT\_MIN) ? cout << "INF " : cout << distances[i] << " ";

}

}

int main() {

int V = 6;

vector<vector<Edge>> graph(V);

graph[0].push\_back({1, 5});

graph[0].push\_back({2, 3});

graph[1].push\_back({3, 6});

graph[1].push\_back({2, 2});

graph[2].push\_back({4, 4});

graph[2].push\_back({5, 2});

graph[2].push\_back({3, 7});

graph[3].push\_back({4, -1});

graph[4].push\_back({5, -2});

int s = 1;

cout << "Đường đi dài nhất từ đỉnh " << s << " là: \n";

longestPath(s, graph, V);

return 0;

}