question 1

## 문제

방향그래프가 주어지면 주어진 시작점에서 다른 모든 정점으로의 최단 경로를 구하는 프로그램을 작성하시오. 단, 모든 간선의 가중치는 10 이하의 자연수이다.

## 입력

첫째 줄에 정점의 개수 V와 간선의 개수 E가 주어진다. (1≤V≤20,000, 1≤E≤300,000) 모든 정점에는 1부터 V까지 번호가 매겨져 있다고 가정한다. 둘째 줄에는 시작 정점의 번호 K(1≤K≤V)가 주어진다. 셋째 줄부터 E개의 줄에 걸쳐 각 간선을 나타내는 세 개의 정수 (u, v, w)가 순서대로 주어진다. 이는 u에서 v로 가는 가중치 w인 간선이 존재한다는 뜻이다. u와 v는 서로 다르며 w는 10 이하의 자연수이다. 서로 다른 두 정점 사이에 여러 개의 간선이 존재할 수도 있음에 유의한다.

## 출력

첫째 줄부터 V개의 줄에 걸쳐, i번째 줄에 i번 정점으로의 최단 경로의 경로값을 출력한다. 시작점 자신은 0으로 출력하고, 경로가 존재하지 않는 경우에는 INF를 출력하면 된다.

answer 1 - 값은 맞으나 배열값이 길게선언되면 런타임 에러로 떠서 실패 ?!??!?!?

import java.util.ArrayList;

import java.util.Scanner;

public class T\_1753 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int edge = scan.nextInt();

int node = scan.nextInt();

int start = scan.nextInt();

//ArrayList<int[]> prev = new ArrayList<int[]>(edge);

int[][] prev = new int[edge][edge];

boolean[] visited = new boolean[edge];

int[] cost = new int[edge];

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

cost[i] = 999998;

}

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

int x = scan.nextInt() - 1;

int y = scan.nextInt() - 1;

prev[x][y] = scan.nextInt();

prev[y][x] = prev[x][y];

}

visited[start - 1] = true;

cost[start - 1] = 0;

int check = start - 1;

int min = 0, out = 0;

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

if (visited[i] == false && (min > prev[start - 1][i] || min == 0)) {

min = cost[i];

check = i;

out = 1;

}

if (prev[start - 1][i] != 0) {

cost[i] = prev[start - 1][i];

}

}

while (true) {

if (out == 0) {

break;

}

out = 0;

min = 999999;

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

if (prev[check][i] != 0 && check != i && !visited[i]) {

if (cost[i] > cost[check] + prev[check][i] || cost[i] == 0) {

cost[i] = cost[check] + prev[check][i];

}

}

}

visited[check] = true;

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

if (visited[i] == false && (min > cost[i] || min == 0)) {

min = cost[i];

check = i;

out = 1;

}

}

}

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

if (cost[i] == 0 && i != start - 1) {

System.out.println("INF");

} else {

System.out.println(cost[i]);

}

}

}

}

answer 2 -