# C. VISUALGO ONLINE QUIZ

## PROBLEM DESCRIPTION

Given a directed weighted graph,
find out the number of possible shortest paths from s to t.

## SOLUTION TECHINQUES

Dijkstra’s Algorithm / Dynamic Programming.

## SOLUTION SKETCHES

An interesting fact in this problem is that: All the weights are greater than 0.
This means there would always be no other ways to get to the vertex with the shortest distance we chose in Dijkstra’s algorithm. Hence we should already know how many shortest paths are there from s to the vertex when it’s selected.

We use p[i] to store the total number of shortest paths from s to i
, and maintain p[i] during the relaxation phase of Dijkstra’s Algorithm.

## TIME COMPLEXITY

O($E \log V$)

## SOLUTION PROGRAM FOR REFERENCE

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <queue>

using namespace std;

const int V = 10005;
const int E = 2e6 + 5;
const int inf = 1e9;

struct vert
{
    int i, d;
    bool operator < (const vert &rhs)const
    {
        return d > rhs.d;
    }
};
struct edge
{
    int to, wt;
};

vector<edge> g[V];
char vst[V];
int d[V];
int p[V];

int main()
{
    int i, v, e, s, t;
    int vi, vj, wt;
    priority_queue<vert> pq;
    scanf("%d%d", &v, &e);
    while (e--)
    {
        scanf("%d%d%d", &vi, &vj, &wt);
        g[vi].push_back( (edge){ vj, wt } );
    }
    fill(d, d + v, inf);
    fill(p, p + v, 0);
    fill(vst, vst + v, 0);
    scanf("%d%d", &s, &t);
    pq.push( (vert){ s, 0 } );
    d[s] = 0;
    p[s] = 1;
    while (!pq.empty())
    {
        vert tmp = pq.top(); pq.pop();
        int &vi = tmp.i;
        if (vst[vi]) continue;
        vst[vi] = 1;
        for (auto &e: g[vi])
        {
            int &vj = e.to;
```

```c
            if (d[vi] + e.wt < d[vj])
            {
                d[vj] = d[vi] + e.wt;
                p[vj] = p[vi];
                pq.push( (vert){ vj, wt } );
            }
            else if (d[vi] + e.wt == d[vj])
                p[vj] += p[vi];
        }
    }
    printf("%d\n", p[t]);
    return 0;
}
```