Your Dijkstra: Shortest Reach 2 submission got 0.00 points. Try Again!

Dijkstra: Shortest Reach 2

by pranav9413

Problem | Submissions | Leaderboard | Discussions | Editorial

Given an undirected graph and a starting node, determine the lengths of the shortest paths from the starting node to all other nodes in the graph. If a node is unreachable, its distance is -1. Nodes will be numbered consecutively from 1 to n, and edges will have varying distances or lengths.

For example, you have the following graph of 5 nodes:

Begin	End	Weight
1	2	5
2	3	6
3	4	2
1	3	15

If we start at node 1, the shortest path to 2 is direct and distance 5. Going from 1 to 3, there are two paths: $1 \rightarrow 2 \rightarrow 3$ at a distance of $5 + 6 = 11$ or $1 \rightarrow 3$ at a distance of 15. We choose the shortest path, 11. From 1 to 4, we choose the shortest path through 3 and extend it: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ for a distance of $11 + 2 = 13$ There is no route to node 5, so the distance is -1.

If we print the distances to all of the nodes in increasing node order and omitting the starting node, we print `5 11 13 -1`.

Input Format

The first line contains t, the number of test cases.

Each test case is as follows:
- The first line contains two space-separated integers n and m, the number of nodes and edges in the graph.
- Each of the next m lines contains three space-separated integers x, y, and r, the beginning and ending nodes of an edge, and the length of the edge.
- The last line of each test case has an integer s, denoting the starting position.

Constraints

$1 \leq t \leq 10$
$2 \leq n \leq 3000$
$1 \leq m \leq \frac{N*(N-1)}{2}$
$1 \leq x, y, s \leq N$
$1 \leq r \leq 10^5$

If there are edges between the same pair of nodes with different weights, they are to be considered as is, like multiple edges.

Output Format

For each of the t test cases, print a single line consisting $n - 1$ space separated integers denoting the shortest distance to the $n - 1$ nodes from starting position s in increasing order of their labels, excluding s.

For unreachable nodes, print -1.

Sample Input

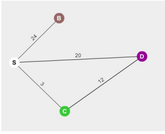
```
1
4 4
1 2 24
1 4 20
3 1 3
4 3 12
1
```

Sample Output

```
24 3 15
```

Explanation

The graph given in the test case is shown as :



- The straight line is a weighted edge, denoting length of edge between the corresponding nodes.
- The nodes S,B,C and D denote the obvious node 1,2,3 and 4 in the test case.

The shortest paths followed for the three nodes B,C and D are as follows :

S->B - Shortest Path Value : 24

S->C - Shortest Path Value : 3

S->C->D - Shortest Path Value : 15

Hard Submitted 16460 times Max Score 60

Need Help?

View Discussions
View Editorial Solution
View Top Submissions

RATE THIS CHALLENGE

Download problem statement
Download sample test cases
Suggest Edits

Current Buffer (saved locally, editable) PHP

```php
<?php

// Complete the shortestReach function below.
function shortestReach($n, $edges, $s) {
    $sOrigem = 0;
    $sConglado = $s;
    $m = count($edges);
    $jumps = $n * $m;
    for($i=0; $i < $m; $i++) {
    //definição de trechos, pontos e distancias
    if(!isset($edges[$i][1])){
        $trajetos[$edges[$i][0].'_0']['distance'] = -1;
        $trajetos[$edges[$i][0].'_0']['verified'] = 1;
    }else $trajetos[$edges[$i][0].'_'.$edges[$i][1]]['distance'] = $edges[$i][2];
    $pontos[] = $edges[$i][0];
    $pontos[] = $edges[$i][1];
    }
    $idPercurso = 1;
    $pontos = array_unique($pontos);
    sort($pontos);
    //contrução dos percursos.
    $jumpsVerify = 0;
    //Montando todos os percursos possíveis.
    do {
      //Construindo todos os trajetos possíveis do percuso.
      do {
        $registry = 0;
        $verify = 0;
        //Percorrendo todos os pontos existentes do percurso.
        foreach ($pontos as $value){
          //procurando os trajetos cadastrados
          if (isset($trajetos[$s.'_'.$value])
            && !isset($trajetos[$s.'_'.$value]['verified'])
            && $s != $sOrigem){
              $rotaDefault = $rota = $s.'_'.$value;
              $registry = 1;
          } elseif (isset($trajetos[$value.'_'.$s])
            && !isset($trajetos[$value.'_'.$s]['verified'])
            && $value != $sOrigem) {
              $rota = $value.'_'.$s;
              $rotaDefault = $s.'_'.$value;
              $registry = 1;
          } else $registry = 0;
          if ($registry) {
              $idPercursoFracao = $idPercurso;
            if ($sOrigem){
              $percursoFracao[$idPercursoFracao] = $percurso[$idPercurso];
              $idPercurso++;
              $percurso[$idPercurso] = $percursoFracao[$idPercursoFracao];
            }
            //registrando da existencia do trecho.
            $trajetos[$rota]['verified'] = 1;
            //adicionando da distancia do trecho ao percurso.
            if (!isset($percurso[$idPercurso]['distance']))
                $percurso[$idPercurso]= array('distance'=>0);
            $percurso[$idPercurso]['distance'] += $trajetos[$rota]['distance'];
            //recuperando a origem e o destino do trecho.
            $startEnd = explode('_', $rotaDefault);
            //atualizando dos pontos percurso A -> B -> C ==> A -> C
            if (empty($percurso[$idPercurso]['start']))
                $percurso[$idPercurso]['start'] = $startEnd[0];
            $percurso[$idPercurso]['finally'] = $startEnd[1];
            //concatenando os trechos percorridos da rota.
            if (!isset($percurso[$idPercurso]['route']))
                $percurso[$idPercurso]['route'] = null;
            $percurso[$idPercurso]['route'] .= $rota.',';
            $s = $startEnd[1];
            $registry = 0;
            $sOrigem = $sConglado;
            break;
          }
          $verify++;
          //Montando todos os percursos possíveis.
          $jumpsVerify++;
        }
      //Verificando os trajetos já traçados.
      } while ($verify < $n);
      //Retornando o valor inicial de $s.
      $s = $sConglado;
      //Determinando $sOrigem como 0 novamente.
      $sOrigem = 0;
      // Incrementando $idPercurso.
      $idPercurso++;
      //Efetuando a verificação dos saltos, que são as vezes que o código percorre as possibilidades de nós.
    } while ($jumpsVerify <= $jumps);
    //Deletando os percursos maiores que apareceram com ['start'] e ['finally'] iguais.
    for($i = 1;$i <= count($percurso); $i++){
      for($j = 1;$j <= count($percurso); $j++){
        if($percurso[$i]['start'] == $percurso[$j]['start']
          && $percurso[$i]['finally'] == $percurso[$j]['finally']
          && $percurso[$i]['distance'] > $percurso[$j]['distance'])
            $percurso[$i]['deleted'] = true;
      }
    }
    //imprimindo o resultado ($result).
    for($i = 1;$i <= count($percurso); $i++)
      if (!isset($percurso[$i]['deleted']))
        $result[] = $percurso[$i]['distance'];
      return $result;
}

$fptr = fopen(getenv("OUTPUT_PATH"), "w");

$stdin = fopen("php://stdin", "r");

fscanf($stdin, "%d\n", $t);

for ($t_itr = 0; $t_itr < $t; $t_itr++) {
    fscanf($stdin, "%[^\n]\n", $nm_temp);
    $nm = explode(' ', $nm_temp);

    $n = intval($nm[0]);

    $m = intval($nm[1]);

    $edges = array();

    for ($i = 0; $i < $m; $i++) {
        fscanf($stdin, "%[^\n]\n", $edges_temp);
        $edges[] = array_map('intval', preg_split('/ /', $edges_temp, -1, PREG_SPLIT_NO_EMPTY));
    }

    fscanf($stdin, "%d\n", $s);

    $result = shortestReach($n, $edges, $s);

    fwrite($fptr, implode(" ", $result) . "\n");
}

fclose($stdin);
fclose($fptr);
```

Line: 101 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Wrong Answer

Ask your friends for help:

✔ Test Case #0	✖ Test Case #1	✖ Test Case #2
⏱ Test Case #3	⏱ Test Case #4	⚠ Test Case #5
⏱ Test Case #6	⚠ Test Case #7	

Try Again