Register for Certification exam (https://examform.nptel.ac.in/2023_01/exam_form/dashboard)

Course outline

How does an NPTEL online course work? ()

Week 0 ()

Week 1 ()

Week 2 ()

Week 3 ()

Week 4 ()

Week 5 ()

Week 6 ()

SSSP - Overview BFS

Week 6: Assignment 6

Your last recorded submission was on 2023-03-08, 14:55 IST Due date: 2023-03-08, 23:59 IST.

1) If all edges have the same weight in an undirected graph, which algorithm will find the shortest path between two nodes more efficiently? ***1 point***

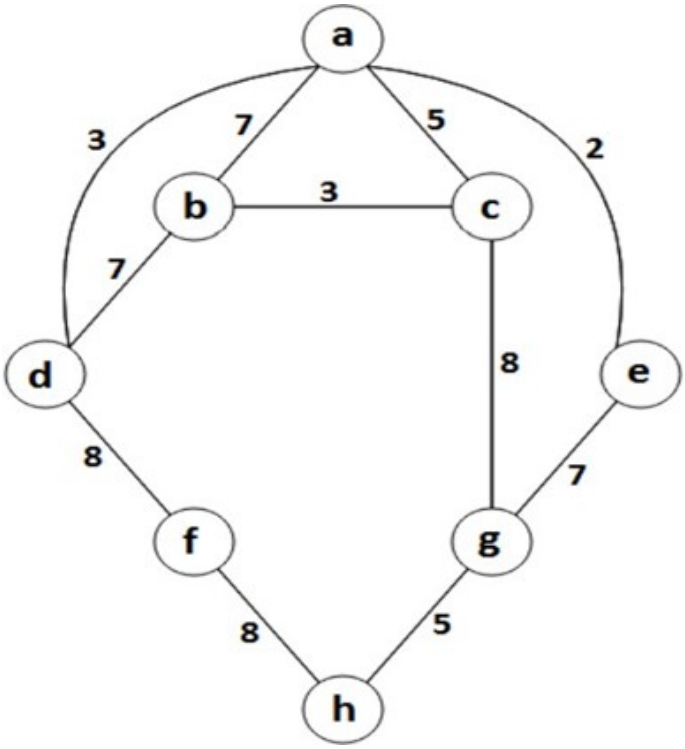
- ○ Dijkstra's algorithm using priority queues
- ○ Bellman-Ford algorithm
- ○ Depth first search
- ◉ Breadth first search

2) Which of the following statement(s) is/are **true** about Dijkstra's algorithm to find shortest path? ***1 point***

- ☑ Dijkstra's algorithm doesn't always work correctly for graphs with negative weights.
- ☐ It returns the shortest path between all pair of nodes.
- ☐ The shortest path returned by Dijkstra's algorithm always passes through the least number of vertices.
- ☑ To decide which node to visit next, Dijkstra's algorithm selects the node with smallest known distance.
- ☐ It can find shortest path in only acyclic graph.

3) In the given graph, if we try to find the shortest path from node `a` to all other nodes using Dijkstra's algorithm, in what order do the nodes get included in the visited set? ***1 point***

Note: *If two vertices have same distance, the algorithm picks the next vertex which comes **first alphabetically***



- (●) a e d c b h f g
- ( ) a e d c b g f h
- ( ) a e d b c g f h
- ( ) a e c d b g f h

4) Which of the following statements is/are true? ***1 point***

I. Given a graph where all edges have positive weights, the shortest path produced by Dijkstra's and Bellman Ford algorithm may be different but path weight would be same.
II. Given a weighted graph where weights of all edges are unique, there is always a unique shortest path from a source to destination in such a graph.

- (●) Only (I)
- ( ) Only (II)
- ( ) Both
- ( ) None of these

5) Consider a weighted, directed acyclic graph $G = (V, E, w)$ in which edges that leave ***1 point*** the source vertex s may have negative weights and all other edge weights are non-negative.

Does Dijkstra's algorithm correctly compute the shortest-path weight $\delta(s, t)$ from s to every vertex t in this graph?

- (●) Yes
- ( ) No

Week 6 Practice Programming Assignment 1 (/noc23_cs30/progassignment?name=206)
Week 6 Practice Programming Assignment 2 (/noc23_cs30/progassignment?name=207)
Week 6 Programming Assignment Q1 (/noc23_cs30/progassignment?name=208)
Week 6 Programming Assignment Q2 (/noc23_cs30/progassignment?name=209)
Week 7 ()
Download Videos ()
Live Sessions ()
Transcripts ()

6) How can we use the Floyd-Warshall algorithm for all-pairs shortest paths to detect whether a graph has a negative cycle? Consider that `SP` is a resultant matrix of Floyd-Warshall algorithm. ***1 point***

- Check if any shortest path entry `SP[i][j]` is negative.
- (selected) Check if any shortest path entry `SP[i][i]` is negative.
- Check if any shortest path entry `SP[i][j]` reduces from one iteration to the next.
- The Floyd-Warshall algorithm cannot be used to detect negative cycles.

Question 7 & 8

Shortest circular route

A traveler made a travel plan which starts from city `S` . Due to time limitations, he decided to choose the shortest circular route that returns to the starting city `S` without using any road twice in the route. The route need not visit all cities. Consider that there is always at least one circular route from the source city.

7) Consider the input in following format:

The first line contains N the number of nodes(represent cities labeled from 0 to N-1) and M the number of undirected edges(represent two way roads between two city).

Next M lines follows, each line contains 3 integers X, Y and Z, which denotes that there is an edge between X and Y with weight Z.

Next line contains 1 integers represent source node.

```
7 11
0 1 10
0 2 50
0 3 300
5 6 45
2 1 30
6 4 37
1 6 65
2 5 76
1 3 40
3 4 60
2 4 20
4
```

What would be total distance of the shortest circular route from source city for given input ?

318

8) Which of the following strategy would work for the given problem? ***1 point***

○

```
AdjList # Adjacency list for graph
def  shortestCircularRoute(AdjList,source_city):
    adjacent_city = []
    for city in AdjList[source_city]:
        adjacent_city.append(city) # add all adjacent city of source city in array

    nearest_adjacent = extract adjacent city with minimum distance from source_city
    dist_edge = distance between nearest_adjacent and source_city
    remove the edge between nearest_adjacent and source_city from AdjList
    dist_path = ShortestPathAlgorithm(AdjList,nearest_adjacent,source_city) #return shortest path length     from nearest_adjacent to source_city

    return(dist_edge+dist_path)
```

◉

```
AdjList # Adjacency list for graph
def  shortestCircularRoute(AdjList,source_city):
    pathweight = []
    adjacent_city = []
    for city in AdjList[source_city]:
        adjacent_city.append(city) # add all adjacent city of source city in array

    for each_city in adjacent_city:
        dist_edge = distance between each_city and source_city
        remove the edge between each_city and source_city from AdjList
        dist_path = ShortestPathAlgorithm(AdjList,each_city,source_city) #return shortest path length           from each_city to source_city
        pathweight.append(dist_edge+ dist_path) # Add circular path length in array
        add the removed edge between each_city and source_city in AdjList
    return(min(pathweight))
```

○

```
AdjList # Adjacency list for graph

def  shortestCircularRoute(AdjList,source_city):
    shortest_length = infinite
    adj = ""
    adjacent_city = []
    for city in AdjList[source_city]:
        adjacent_city.append(city) # add all adjacent city of source city in
array

    for each_city in adjacent_city:
        remove the edge between each_city and source from AdjList
        dist_path  =  ShortestPathAlgorithm(AdjList,each_city,source_city)
#return shortest path length           from each_city to source_city
        if dist_path < shortest_length:
            shortest_length = dist_path
            adj = each_city
        add the removed edge between each_city and source_city in AdjList

    dist_edge = distance between adj and source_city
    return(shortest_length + dist_edge)
```

You may submit any number of times before the due date. The final submission will be considered for grading.

Submit Answers