

Advanced Algorithms & Data Structures



Department of CSE

ADVANCED ALGORITHMS AND DATA STRUCTURES 23CS03HF

Topic:

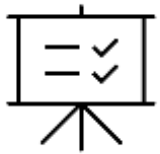
Single Source Shortest Path Problem

AIM OF THE SESSION



To familiarize students with the concept of Single Source Shortest Path Problem

INSTRUCTIONAL OBJECTIVES



This Session is designed to:

1. Demonstrate :- Single Source Shortest Path Problem.
2. Describe :- solve single-source shortest path problem using Dijkstra's Algorithm.

LEARNING OUTCOMES



At the end of this session, you should be able to:

1. Define :- Single Source Shortest path Problem.
2. Describe :- solve single-source shortest path problem using Dijkstra's Algorithm
3. Summarize:- Finding the shortest paths from a single source vertex to all other vertices in a weighted graph.

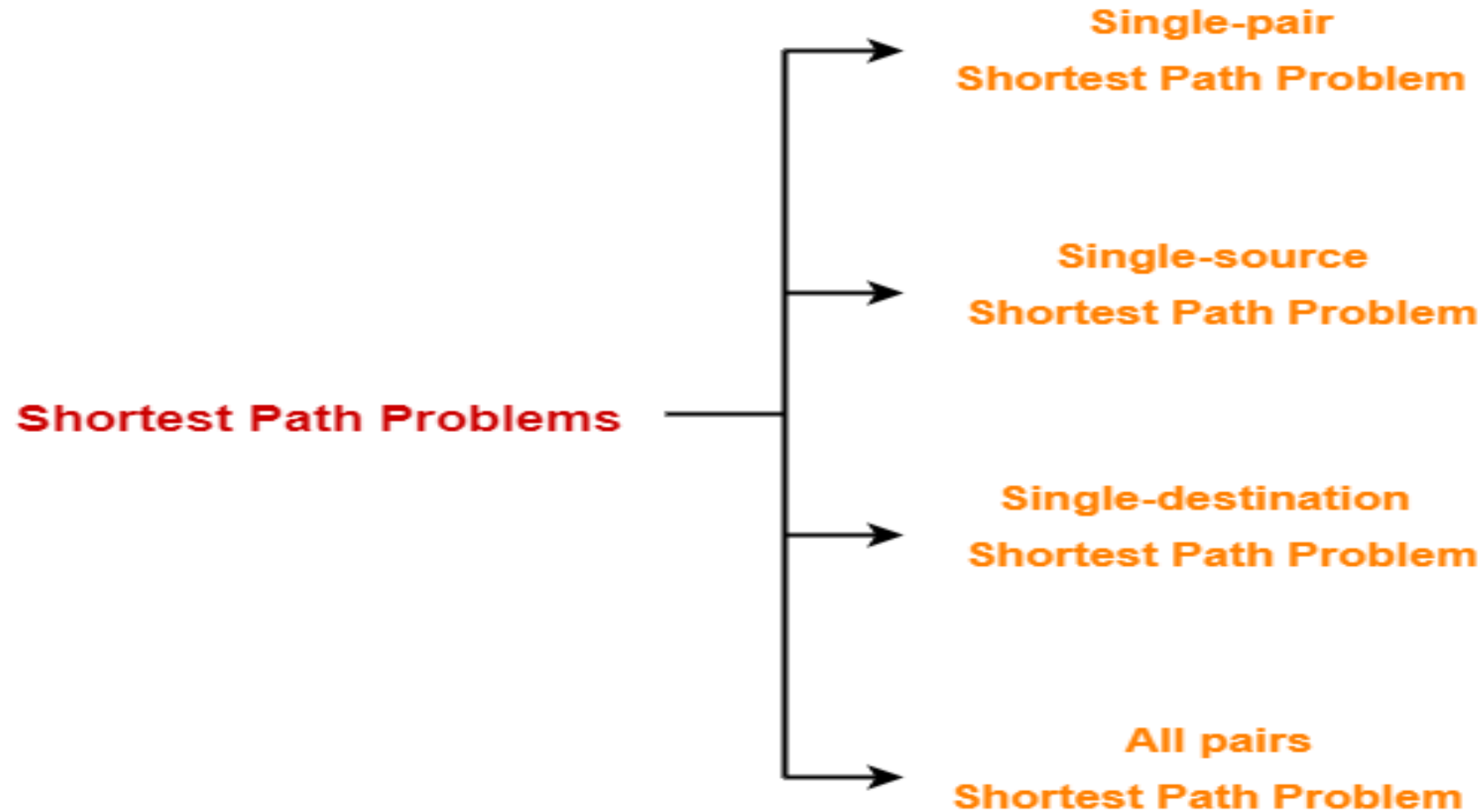
Shortest Path Problem

- Shortest path problem is a problem of finding the shortest path(s) between vertices of a given graph.
- Shortest path between two vertices is a path that has the least cost as compared to all other existing paths.

Applications

- Google Maps
- Road Networks
- Logistics Research

Types of Shortest Path Problem-



Single Source Shortest Path Problem

- It is a shortest path problem where the shortest path from a given source vertex to all other remaining vertices is computed.
- **Dijkstra's Algorithm** and Bellman Ford Algorithm are the famous algorithms used for solving single-source shortest path problem.

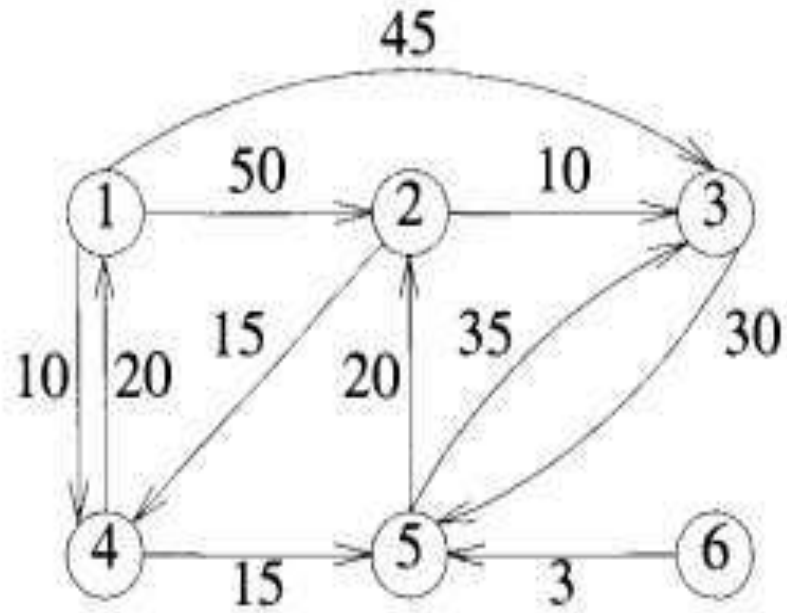
Dijkstra Algorithm

- Dijkstra Algorithm is a very famous greedy algorithm.
- It is used for solving the single source shortest path problem.
- It computes the shortest path from one particular source node to all other remaining nodes of the graph.

Conditions

It is important to note the following points regarding Dijkstra Algorithm-

- Dijkstra algorithm works only for **connected graphs**.
- Dijkstra algorithm works only for those graphs that **do not contain any negative weight edge**.
- The actual Dijkstra algorithm **does not output the shortest paths**.
- It only provides the **value or cost of the shortest paths**.
- Dijkstra algorithm works for **directed as well as undirected graphs**.



(a) Graph

<i>Path</i>	<i>Length</i>
1) 1, 4	10
2) 1, 4, 5	25
3) 1, 4, 5, 2	45
4) 1, 3	45

(b) Shortest paths from 1

Implementation

Source Vertex	Intermediate Vertex	dist[2]	dist[3]	dist[4]	dist[5]	dist[6]	Path	Distance
1	-	50	45	<u>10</u>	∞	∞	1 \rightarrow 4	10
	4	50	45	-	<u>25</u>	∞	1 \rightarrow 4 \rightarrow 5	25
	5	<u>45</u>	45	-	-	∞	1 \rightarrow 4 \rightarrow 5 \rightarrow 2	45
	2	-	<u>45</u>	-	-	∞	1 \rightarrow 3	45
	3	-	-	-	-	∞	No path to 6	

```
1  Algorithm ShortestPaths( $v, cost, dist, n$ )
2  //  $dist[j]$ ,  $1 \leq j \leq n$ , is set to the length of the shortest
3  // path from vertex  $v$  to vertex  $j$  in a digraph  $G$  with  $n$ 
4  // vertices.  $dist[v]$  is set to zero.  $G$  is represented by its
5  // cost adjacency matrix  $cost[1 : n, 1 : n]$ .
6  {
7      for  $i := 1$  to  $n$  do
8      { // Initialize  $S$ .
9           $S[i] := \text{false}$ ;  $dist[i] := cost[v, i]$ ;
10     }
11      $S[v] := \text{true}$ ;  $dist[v] := 0.0$ ; // Put  $v$  in  $S$ .
12     for  $num := 2$  to  $n - 1$  do
13     {
14         // Determine  $n - 1$  paths from  $v$ .
15         Choose  $u$  from among those vertices not
16         in  $S$  such that  $dist[u]$  is minimum;
17          $S[u] := \text{true}$ ; // Put  $u$  in  $S$ .
18         for (each  $w$  adjacent to  $u$  with  $S[w] = \text{false}$ ) do
19             // Update distances.
20             if ( $dist[w] > dist[u] + cost[u, w]$ ) then
21                  $dist[w] := dist[u] + cost[u, w]$ ;
22     }
23 }
```

Time Complexity Analysis-

- The given graph G is represented as an adjacency matrix.
- Priority queue Q is represented as an unordered list.

Here,

- $A[i,j]$ stores the information about edge (i,j) .
- Time taken for selecting i with the smallest dist is $O(V)$.
- For each neighbor of i , time taken for updating $dist[j]$ is $O(1)$ and there will be maximum V neighbors.
- Time taken for each iteration of the loop is $O(V)$ and one vertex is deleted from Q .
- Thus, total time complexity becomes $O(V^2)$.

- **Objective:** Find the shortest paths from a single source vertex to all other vertices in a weighted graph.
- **Approach:** Initialize distances from the source to all vertices as infinity, update distances using relaxation to minimize path weights, and repeat until all shortest paths are found.
- **Algorithmic Solutions:** Algorithms like Dijkstra's (for non-negative weights) used to solve this problem efficiently.

SELF-ASSESSMENT QUESTIONS

Which of the following conditions must be true for Dijkstra's algorithm to work correctly?

- (a) The graph must be directed.
- (b) The graph must be undirected.
- (c) The graph must have non-negative edge weights.
- (d) The graph must have no cycles.

What is the main application of single source shortest path algorithms?

- (a) Finding Minimum Spanning Tree
- (b) Finding shortest paths from a single source node to all other nodes
- (c) Detecting cycles in a graph
- (d) Sorting nodes in topological order

TERMINAL QUESTIONS

1. Can you explain the basic architecture of Dijkstra's algorithm along with an example?
2. What are some real-world scenarios where Dijkstra's algorithm can be applied effectively?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books :

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein., 3rd, 2009, The MIT Press.
- 2 Algorithm Design Manual, Steven S. Skiena., 2nd, 2008, Springer.
- 3 Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser., 2nd, 2013, Wiley.
- 4 The Art of Computer Programming, Donald E. Knuth, 3rd, 1997, Addison-Wesley Professiona.

MOOCS :

1. <https://www.coursera.org/specializations/algorithms?=>
2. <https://www.coursera.org/learn/dynamic-programming-greedy-algorithms#modules>

THANK YOU

