**Data Structures Lab**

***Session 13***

**Course:** Data Structures (CL-2001) **Semester:** Fall 2021

**Instructor:**  **T.A:** N/A

**Note:**

* + - * Lab manual cover following topics

**{Application of Graph Data Structures , Minimum Spanning Tree, Shortest Path and All Pair Shortest Path using Graph}**

* Maintain discipline during the lab.
* Just raise your hand if you have any problem.
* Completing all tasks of each lab is compulsory.
* Get your lab checked at the end of the session.

**Application of Graph Data Structures**

1. In **Computer science** graphs are used to represent the flow of **computation**.
2. **Google maps** uses graphs for building transportation systems, where intersection of two(or more) roads are considered to be a vertex and the road connecting two vertices is considered to be an edge, thus their navigation system is based on the algorithm to calculate the shortest path between two vertices.
3. In **Facebook**, users are considered to be the vertices and if they are friends then there is an edge running between them. Facebook’s Friend suggestion algorithm uses graph theory. Facebook is an example of an undirected graph.
4. In **World Wide Web**, web pages are considered to be the vertices. There is an edge from a **page** **u** to other **page v** if there is a link of page v on page u. This is an example of Directed graph. It was the basic idea behind Google Page Ranking Algorithm.
5. In **Operating System**, we come across the Resource Allocation Graph where each process and resources are considered to be vertices. Edges are drawn from resources to the allocated process, or from requesting process to the requested resource. If this leads to any formation of a cycle then a deadlock will occur.

For Applications of Graph Data Structure please follow link: <https://leapgraph.com/graph-data-structures-applications/>

**Minimum Spanning Tree**

A minimum spanning tree is a spanning tree in which the sum of the weight of the edges is as minimum as possible.

| Weighted Graph | A possible spanning tree |
| --- | --- |

The minimum spanning tree from a graph is found using the following algorithms:

1. Prim's Algorithm
2. Kruskal's Algorithm

**Prim’s Algorithm**

Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

1. form a tree that includes every vertex
2. has the minimum sum of weights among all the trees that can be formed from the graph
3. Prim’s Algorithm is a famous greedy algorithm.
4. It is used for finding the Minimum Spanning Tree (MST) of a given graph.
5. To apply Prim’s algorithm, the given graph must be weighted, connected and undirected.

**How Prim's algorithm works**

The implementation of Prim’s Algorithm is explained in the following steps-

**Step-01:**

* Randomly choose any vertex.
* The vertex connecting to the edge having least weight is usually selected.

**Step-02:**

* Find all the edges that connect the tree to new vertices.
* Find the least weight edge among those edges and include it in the existing tree.
* If including that edge creates a cycle, then reject that edge and look for the next least weight edge.

**Step-03:**

* Keep repeating step-02 until all the vertices are included and Minimum Spanning Tree (MST) is obtained.

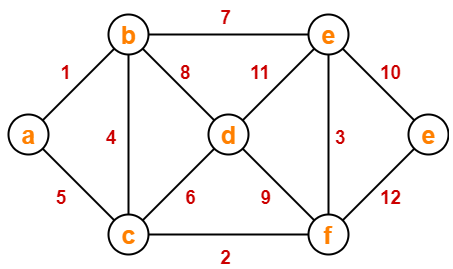
**Example:**

Please follow thee following link:

<https://www.youtube.com/watch?v=ZtZaR7EcI5Y&t=290s>

**Task-1**

1. **Dry Run the below graph starting with Node a.**
2. **Implement the above dry run code using graph class which we built in LAb 12.**

****

**Kruskal’s Algorithm**

Kruskal's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

1. form a tree that includes every vertex
2. has the minimum sum of weights among all the trees that can be formed from the graph
3. Kruskal’s Algorithm is a famous greedy algorithm.
4. It is used for finding the Minimum Spanning Tree (MST) of a given graph.
5. To apply Kruskal’s algorithm, the given graph must be weighted, connected and undirected.

**How Kruskal's algorithm works**

The implementation of Kruskal’s Algorithm is explained in the following steps-

**Step-01:**

* Sort all the edges from low weight to high weight.

**Step-02:**

* Take the edge with the lowest weight and use it to connect the vertices of graph.
* If adding an edge creates a cycle, then reject that edge and go for the next least weight edge.

**Step-03:**

* Keep adding edges until all the vertices are connected and a Minimum Spanning Tree (MST) is obtained.

**Example:**

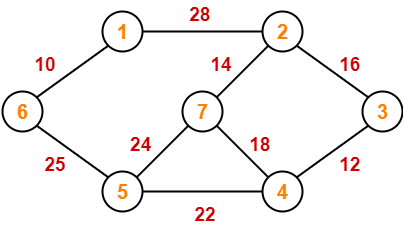
Please follow thee following link:

<https://www.youtube.com/watch?v=EjVHtpWkIho&t=490s>

**Task-2**

**1. Dry Run the below graph and find the minimum spanning tree.**

**2. Implement the above dry run code using the graph class which we built in LAb 12.**

****

**Shortest Path and All Pair Shortest Path**

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

**Types of Shortest Path Problem:**

Various types of shortest path problem are but we will discuss about two of them:

1. Single-source shortest path problem
2. All pairs shortest path problem

**Single source shortest path**

* 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’s 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.
* By making minor modifications in the actual algorithm, the shortest paths can be easily obtained.
* Dijkstra algorithm works for directed as well as undirected graphs.

**How Dijkstra’s algorithm works**

The implementation of above Dijkstra Algorithm is explained in the following steps:

1. **S**et all vertices distances = infinity except for the source vertex, set the source distance = 0.
2. Push the source vertex in a min-priority queue in the form (distance , vertex),as the comparison in the min-priority queue will be according to vertices distances.
3. Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).
4. Update the distances of the connected vertices to the popped vertex in case of "current vertex distance + edge weight < next vertex distance", then push the vertex with the new distance to the priority queue.
5. If the popped vertex is visited before, just continue without using it.
6. Apply the same algorithm again until the priority queue is empty.

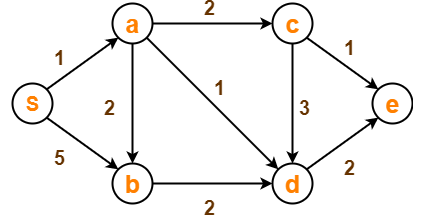
**Example:**

Please follow thee following link:

<https://www.youtube.com/watch?v=smHnz2RHJBY&t=1816s>

**Task-3**

1. **Dry Run the below graph and find the minimum spanning tree.**
2. **Implement the below dry run code using the graph class which we built in LAb 12.**

****

**All pair shortest path**

* It is a shortest path problem where the shortest path between every pair of vertices is computed.
* **Floyd-Warshall Algorithm** and **Johnson’s Algorithm** are the famous algorithms used for solving All pairs shortest path problem.

## Floyd Warshall Algorithm:

* Floyd Warshall Algorithm is a famous algorithm.
* It is used to solve All Pairs Shortest Path Problem.
* It computes the shortest path between every pair of vertices of the given graph.
* Floyd Warshall Algorithm is an example of a dynamic programming approach.
* Floyd Warshall Algorithm is best suited for dense graphs.

**How Floyd Warshall works:**

**Create a |V| x |V| matrix // It represents the distance between every pair of vertices as given**

**For each cell (i,j) in M do-**

**if i = = j**

**M[ i ][ j ] = 0 // For all diagonal elements, value = 0**

**if (i , j) is an edge in E**

**M[ i ][ j ] = weight(i,j) // If there exists a direct edge between the vertices, value = weight of edge**

**else**

**M[ i ][ j ] = infinity // If there is no direct edge between the vertices, value = ∞**

**for k from 1 to |V|**

**for i from 1 to |V|**

**for j from 1 to |V|**

**if M[ i ][ j ] > M[ i ][ k ] + M[ k ][ j ]**

**M[ i ][ j ] = M[ i ][ k ] + M[ k ][ j ]**

**Example:**

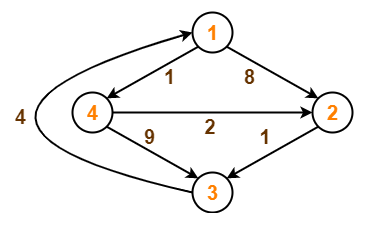
Please follow thee following link:

<https://www.youtube.com/watch?v=Gc4mWrmJBsw>

<https://www.gatevidyalay.com/floyd-warshall-algorithm-shortest-path-algorithm/>

**Task-4**

* **Dry Run the above task with changing the configuration of figure-1 starting with** **Node 3**
* **Implement the above dry run code**

****

| **Lab13: Application of Graph Data Structures , Minimum Spanning Tree, Shortest**  **Path and All Pair Shortest Path using Graph** | | |
| --- | --- | --- |
| **Std Name: Std\_ID:** | | |
|  | | |
| **Lab1-Tasks** | **Completed** | **Checked** |
| Task #1 |  |  |
| Task #2 |  |  |
| Task #3 |  |  |
| Task# 4 |  |  |