CSN-261

Data structure and laboratory

By: Kanhaiya kumar

Enrollment no. 19114042

Problem -1

Write a C++ program to implement a graph using adjacency list (linked list) without using STL. Perform following operations on the graph after creating the graph based on the edge list given in the input file.

1. BFS traversal
2. DFS traversal
3. Cycle finding in the graph
4. Calculate diameter of the graph

Problem -1

Data structure: Considering above problem,I implemented the same with array of linked list (to create adjacency list).There is a linked list defined for each vertex that stores its adjacent neighbours.I have implemented linked list using struct in my solution.

Algorithm: Various algorithm were used for solving subparts of this problem.DFS algorithm was used for subpart 2 and 3.DFS algorithm can be used for a number of applications including finding path between 2 node,Detecting cycles in graph ,for testing bipartite graph etc.

Pseudo code for DFS:

```
DFS(G,V)
mark V as visited
for each adjacent_vertices U of V:
    if U is not visited
```

Problem -1

```
DFS(G,U)
MAIN(){
        G is graph
        For each u ∈ G
      make u as unvisited
    For each u ∈ G
      DFS(G, u)
}
```

Time complexity(DFS): Time complexity for DFS algorithm is O(V+E), where V is total number of vertices and E is total number of edges.

space complexity(BFS): Space complexity is O(V)

Another algorithm used was BFS(Breadth first search algorithm).Application of BFS includes finding maximum flow in a network,Finding minimum spanning tree,Cycle detection in undirected graph etc.

Problem -1

Pseudo code (BFS):

create a queue Q
mark vertex v as visited and put v into Q
while Q is non-empty
    pop-up a vertex from Q
    mark and enqueue all (unvisited) neighbours of u

Time complexity(BFS): Time complexity for BFS algorithm is O(V+E), where V is total number of vertices and E is total number of edges which is same as that of DFS.

space complexity(BFS): Space complexity is O(V)

Screenshot problem-1

Custom Input

```
1 4
2 5
2 3
3 5
3 6
3 7
```

Status Successfully executed **Date** 2020-10-21 13:37:31 **Time** 0 sec **Mem** 5.972 kB

Input

```
7
1 2
1 4
2 5
2 3
3 5
```

Output

```
1. 1 4 2 3 5 7 6
2. 1 4 2 3 7 6 5
3. Cycle: Yes
4. Diameter: 4
```

Problem-2

Given a set of nodes connected to each other in the form of a weighted undirected graph G, find the minimum spanning tree (MST). A spanning tree T of an undirected graphG is a subgraph that is a tree which includes all of the vertices of G, with minimum possible number of edges. G may have more than one spanning trees. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree. A minimum spanning tree (MST) is a spanning tree whose weight is less than or equal to that of every other spanning tree.

Data structure: As specified in problem statement, UNION-FIND data structure was used to solve this problem.

Algorithm: Kruskal's algorithm was used for solving this problem.This algorithm is mainly used to solve networking problems in real life.Ex: LAN connection.

Problem-2

Pseudo code for Kruskal's algorithm:

```
        KRUSKAL(G):
    A = ∅
    For each vertex v ∈ Graph:
        MAKE-SET(v)
    For each edge (u, v) ∈ Edge(G) ordered in ascending  order by weight(u, v):
        if FIND-SET(u) ≠ FIND-SET(v):
        A = A U {(u, v)}
        UNION(u, v)
    return A
```

Time complexity :Time complexity for Kruskal's algorithm is O(ElogE), where E is total number of edges in graph.

Screenshot-Problem 2

Custom Input

```
3 4 3
3 6 2
3 5 4
4 5 3
6 5 3
```

Status Successfully executed **Date** 2020-10-21 11:22:00 **Time** 0 sec **Mem** 5.972 kB

Input

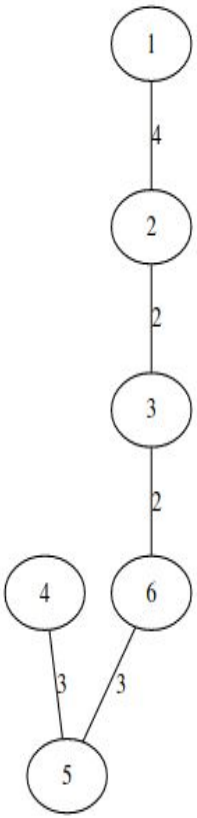
```
1 2 4
1 3 4
2 3 2
3 4 3
3 6 2
3 5 4
```

Output

```
Node1  Node2  Weight
3      6      2
2      3      2
6      5      3
4      5      3
1      3      4
```

MST for problem 2

```
graph mst{
    overlap = "scale";
    2 -- 3 [label = "2"];
    3 -- 6 [label = "2"];
    6 -- 5 [label = "3"];
    4 -- 5 [label = "3"];
    1 -- 2 [label = "4"];
}
```



The End