National Institute of Technology Calicut

Department of Computer Science and Engineering

Second Semester MCA

CS2094D Data Structures Laboratory

Assignment 4, Submission Date: 19.05.2022

Policies for Submission and Evaluation

You must submit your assignment in the Moodle (Eduserver) course page, on or before the submission deadline. Also, ensure that your programs in the assignment must compile and execute without errors in any server. Failure to execute programs in the assignment without compilation errors may lead to zero marks for that program. Your submission will also be tested for plagiarism, by automated tools. In case your code fails to pass the test, you will be straightaway awarded zero marks for this assignment.

Naming Conventions for Submission

Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar or .tar.gz).The name of this file must be $Firstname\_Regno\_Assg\#(e.g., John\_m200034ca\_Assg4.Zip)$. DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.

Question 1

A disjoint-set data structure maintains a collection S = S1, S2. . . .Sk of disjoint dynamic sets. Identify each set by a representative, which is some member of the set. Write a program that implements the Disjoint-set data structure using rooted forests. Also, write functions to implement the ranked union and path compression heuristics on your data structure.

MAKESET(x) : creates a new set whose only member (and thus representative) is x. Since the sets are disjoint, we require that x not already be in some other set.

FIND(x) : finds the representative of the set containing the element x.

UNION(x, y) : unites the dynamic sets that contain x and y, say S_x and S_y, into a new set that is the union of these two sets. The representative of the resultant set is assigned with FIND(x), unless the ranked union heuristic is used and the ranks of both FIND(x) and FIND(y) are different. Otherwise, the representative is assigned in accordance with the ranked union heuristic.

Note that looking up an element in the data structure must be done in O(1) time.

Input Format

- The input consists of multiple lines, each one containing a character from 'm', 'f', 'u', 'e' followed by zero, one or two integers separated by a single space. The integer(s), if given, is in the range 0 to 10000.

- Call the function MAKESET(x) if the input line contains the character ‘m’ followed by an integer x. Print -1 if x is already present in some set, and the value of x, otherwise.
- Call the function FIND(x) if the input line contains the character ‘f’ followed by an integer x. Print the value of find(x) if x is present, and -1 if x is not present.
- Call the function UNION(x,y) if the input line contains the character ‘u’ followed by space-separated integers x and y. Print -1, if either x or y isn’t present in the disjoint set. Print FIND(x) itself if FIND(x)=FIND(y). Otherwise, print the representative of the resultant set. The representative of the resultant set is assigned with FIND(x), unless the ranked union heuristic is used and the ranks of both FIND(x) and FIND(y) are different. Otherwise, the representative is assigned in accordance with the ranked union heuristic.
- If the input line contains the character ‘e’, terminate the program.

Output Format

The output of FIND(x) and UNION(x) consists of multiple space separated values. The values correspond to the following disjoint-set data structures:

- with neither ranked union nor path compression applied.
- with only ranked union applied.
- with only path compression applied.

Sample Input and Output

Input

```
m 1
m 2
m 3
m 4
m 5
m 6
m 7
m 8
m 9
u 1 2
u 3 4
u 5 6
u 7 8
u 9 8
u 6 8
u 4 8
u 2 8
f 9
m 10
u 10 9
e
```

Output

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

```
8
9
1 1 1
3 3 3
5 5 5
7 7 7
9 7 9
5 5 5
3 5 3
1 5 1
1 5 1
10
10 5 10
```

Question 2

Write a program to find the adjacency list of a given directed graph G which is represented as adjacency matrix.

Input format

- The first line of the input contains a positive integer n, the number of vertices in the graph, in the range 1 to 1000.
- The next line represents the Adjacency matrix representation of the given graph.

Output Format:

- The first n lines contain the adjacency list of each node in ascending order.
- Each line contains the label of the respective node followed by the nodes adjacent to it sorted in ascending order from left to right separated by a space.
- If a node has no adjacent nodes, then the line corresponding to its adjacency list will contain the label of that node only.

Note: In a graph with n vertices, the vertices are labeled from 0 to n 1.

Sample Input and Output
Input:

```
5
0 1 0 0 1
0 0 1 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 1 0
```

Output:

```
0 1 4
1 2
2
3 2
4 3
```

Question 3

Write a program to implement following graph search algorithms in a directed graph. Assume that the vertex ordering in a graph follows the natural number sequence starting from 0.

- Breadth First Search (BFS)
- Depth First Search (DFS)

Input format

- First line contains two integers $n \in [1, 1000]$ and $m \in [1, 1000]$ denoting the number of vertices and edges respectively.
- Next m lines denote the pair of vertices representing the edge.
- The last line contains the source vertex.

Output Format:

- Print the BFS and DFS traversal respectively in two different lines.

Sample Input and Output
Input:

```
4 6
0 1
0 2
1 2
2 0
2 3
3 3
0
```

Output:

```
0 1 2 3
0 1 2 3
```

Question 4

Write programs to compute the minimum spanning tree of a connected undirected graph G using the following algorithms: .

a. Kruskal's algorithm
b. Prim's algorithm

Input format

- First line contains a character from 'a', 'b'
  - If the input character is 'a' then compute the minimum spanning tree using Kruskal's algorithm
  - Else if the character is 'b' compute the minimum spanning tree using Prim's algorithm
- Second line contains an integer $n \in [1, 1000]$, that denotes the number of vertices in the graph.
- The subsequent n lines contain the label of the respective node followed by the nodes adjacent to it sorted in ascending order from left to right separated by a space.

- The subsequent n lines contain label of the respective node followed by the weights of the edges corresponding to the adjacency list separated by a space. The edge weights are real numbers in the range [-10000, 10000]. Further, no two edges have the same weight.

Output Format:

- Single line containing the sum of the edge weights of the minimum spanning tree.

Note:

- In a graph with n vertices, the vertices are labeled from 0 to n 1.
- Use **adjacency lists** to store the graphs, with the vertices sorted in ascending order. The adjacency list of each node is a singly linked list that contains its adjacent nodes sorted in ascending order from left to right. The nodes in this list contain two fields, namely, the label of the adjacent node and the weight of the edge, if provided. Unless specified otherwise, the adjacency lists must be processed iteratively from left to right.

Sample Input and Output

Input1:

```
a
7
0 1 5
1 0 2 6
2 1 3
3 2 4 6
43 5 6
5 0 4
6 1 3 4
0 28 10
1 28 16 14
2 16 12
3 12 22 18
4 22 25 24
5 10 25
6 14 18 24
```

Output 1:

```
99
```

Input2:

```
b
7
0 1 5
1 0 2 6
2 1 3
3 2 4 6
4 3 5 6
5 0 4
6 1 3 4
0 28 10
1 28 16 14
2 16 12
3 12 22 18
4 22 25 24
5 10 25
6 14 18 24
```

Output 2:

```
99
```

Question 5

Write a program to check if there is a negative cycle in a directed graph. A negative cycle is one in which the overall sum of the weights in the cycle is negative.

Input format

- The first line contains two integers $n \in [1, 1000]$ and $m \in [1, 1000]$ denoting the number of vertices and number of edges present in a directed graph, respectively.
- Next m lines contain three integers x, y, and w denoting the directed edge from x to y having a weight equal to w.

Output Format:

- Print 1 if there is a negative cycle. Otherwise, print -1.

Note:

- The vertices are labeled from 0 to n-1.

Sample Input and Output

Input1:

```
4 4
0 1 1
1 2 -1
2 3 -1
3 0 -1
```

Output 1:

```
1
```

Input2:

```
5 8
0 1 -1
0 2 4
1 2 3
1 3 2
1 4 2
3 2 5
3 1 1
4 3 -3
```

Output 2:

```
-1
```