National Institute of Technology, Calicut
Department of Computer Science and Engineering
CS2094 – Data Structures Lab

Assignment 5

Date of Submission: On or before 19-04-2015(Sunday) 5:00pm
(for both Advanced batch and Main batch)

Policies for Submission and Evaluation

You must submit your assignment in the moodle (eduserver) course page, on or before the submission deadline.

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 F grade in the course. Detection of ANY malpractice regarding the lab course will also may lead to awarding an F grade.

Naming Conventions for submission

The source codes must be named as ASSG<Number>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.<extension> (For example: ASSG5_BxxyyyyCS_LAXMAN_1.c). If there is a part a and a part b for a particular question, then, name the source files for each part separately as in ASSG5_BxxyyyyCS_LAXMAN_1b.cpp

If you do not conform to the above naming conventions, your submission might not be recognized by some automated tools, and hence will lead to a score of 0 for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

Violations of academic integrity will be severely penalized.
Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work **MUST BE** an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign **F** grade in the course. The department policy on academic integrity can be found at: http://cse.nitc.ac.in/sites/default/files/Academic-Integrity.pdf.

About the Assignment

This assignment is based on disjoint sets and graphs. Implement ALL questions. Your program MUST take input from text files, which is formatted as specified with each question.

Questions

1. **DISJOINT**
   Implement the disjoint-set data structure using disjoint-set forests (rooted trees).
   a) Without ranked union and without path compression
   b) Without ranked union and with path compression
   c) With ranked union and without path compression
   d) With ranked union and with path compression

   Your program must support the following functions:
   - **makeset(x)** – creates a set with one element whose data is specified by x.
   - **find(x)** – finds the set (representative) to which the data specified by x belongs.
   - **union(x, y)** – merges the sets containing the data specified by x and y together, into a single set. After this operation, x and y (as well as the other data that were also contained in the sets which contained x and y) will belong to the same set.

   Input - Output Format
   - The input consists of multiple lines, each one containing either one or two integers.
   - The first integer in the line can be 0, 1, 2 or 3, and each one has its own meaning:
   - The integer 0 means stop the program.
   - The integer 1 stands for the makeset operation. The data for makeset will be the next integer from the input.
   - The integer 2 stands for the find operation. Output the representative of the set, which contains the next integer from the input. (You are guaranteed that the input data will be contained in some set).
   - The integer 3 stands for the union operation. Merge the sets containing each of the next two input integers into a single set.

```
Sample Input  Sample Output (with ranked union and path compression)
1 25
1 35
1 45
1 55
2 35          35
2 45          45
3 35 45
2 35          35
2 45          35
2 25          25
3 25 45
2 25          35
3 45 25
2 25          35
2 45          35
```

<u>GRAPHS</u>

<u>Common Input Format</u>

The first line of the input file contains a positive integer **N**, the number of vertices in the graph. The set of vertices, **V** contains vertices that are labeled **0**, **1**, **2**, …, **N-1**
The second line contains a non-negative integer **E**, the number of edges in the graph. (Assume undirected graph).
Then, **E** lines follow, each one containing three space-separated integers **u**, **v** and **c**
(**u**, **v** ∈ **V**). This means that there is an undirected edge that connects the vertices labeled **u** and **v**, and the cost of this edge is **c**.

Note: There can be more than one different correct output for a question. The sample output given here contains only one of them.

<u>Sample Input</u>

```
12
20
0 9 41
0 8 27
1 2 10
1 3 11
1 4 17
2 3 7
2 5 33
2 6 44
3 4 26
4 5 5
4 7 8
4 8 15
4 9 16
5 6 21
6 7 31
6 10 18
6 11 29
7 8 20
8 9 13
10 11 23
```

<u>Questions</u>

2. **BFS**

Write a program that performs Breadth First Search in a graph.

Take an integer from the terminal as an extra input. Do the BFS, starting from this vertex.

Output Format

The output must contain exactly N integers on a single line – the sequence of vertex labels, in the order they are visited.

Sample Output (start at 1)

```
1 2 3 4 5 6 7 8 9 10 11 0
```

3. **DFS**

   Write a program that performs Depth First Search in a graph.
   Take an integer from the terminal as an extra input. Do the DFS, starting from this vertex.

   Output Format

   The output must contain exactly N integers on a single line – the sequence of vertex labels, in the order they are visited.

   Sample Output (start at 1)

   ```
   1 2 3 4 5 6 7 8 0 9 10 11
   ```

4. **DIJKSTRA**

   Write a program that implements Dijkstra's algorithm.
   Take an integer from the terminal as an extra input. This is the source vertex for Dijkstra's algorithm.

   Output Format

   The output must contain exactly N integers on a single line – the list of lengths of shortest paths from the source vertex (in the order of labels of vertices – 0 to N-1).

   Sample Output (src node = 1)

   ```
   59 0 10 11 17 22 14 25 32 33 32 43
   ```

5. **KRUSKAL**

   Write a program that implements Kruskal's algorithm.

   Output Format

   Output the cost of the minimum spanning tree on the first line. Below that, output the adjacency matrix of the tree.

Sample Output

```
164
 0  0  0  0  0  0  0  0 27  0  0  0
 0  0 10  0 17  0  0  0  0  0  0  0
 0 10  0  7  0  0  0  0  0  0  0  0
 0  0  7  0  0  0  0  0  0  0  0  0
 0 17  0  0  0  5  0  8 15  0  0  0
 0  0  0  0  5  0 21  0  0  0  0  0
 0  0  0  0  0 21  0  0  0  0 18  0
 0  0  0  0  8  0  0  0  0  0  0  0
27  0  0  0 15  0  0  0  0 13  0  0
 0  0  0  0  0  0  0  0 13  0  0  0
 0  0  0  0  0  0 18  0  0  0  0 23
 0  0  0  0  0  0  0  0  0  0 23  0
```

6. PRIM

Write a program that implements Prim’s algorithm.

Output Format

Output the cost of the minimum spanning tree on the first line. Below that, output the adjacency matrix of the tree.

Sample Output

```
164
 0  0  0  0  0  0  0  0 27  0  0  0
 0  0 10  0 17  0  0  0  0  0  0  0
 0 10  0  7  0  0  0  0  0  0  0  0
 0  0  7  0  0  0  0  0  0  0  0  0
 0 17  0  0  0  5  0  8 15  0  0  0
 0  0  0  0  5  0 21  0  0  0  0  0
 0  0  0  0  0 21  0  0  0  0 18  0
 0  0  0  0  8  0  0  0  0  0  0  0
27  0  0  0 15  0  0  0  0 13  0  0
 0  0  0  0  0  0  0  0 13  0  0  0
 0  0  0  0  0  0 18  0  0  0  0 23
 0  0  0  0  0  0  0  0  0  0 23  0
```

****