National Institute of Technology Calicut
Department of Computer Science and Engineering
Fourth Semester B. Tech.(CSE)-Winter 2021
CS2094D Data Structures Laboratory
Assignment #4

Submission deadline (on or before): 22.04.2021, 12:00 PM

Policies for Submission and Evaluation:

- Programs should be written in C language and compiled using C compiler in Linux platform.
- Ensure that your programs will compile and execute without errors in the Linux platform.
- During the evaluation, failure to execute programs without compilation errors may lead to zero marks for that evaluation.
- 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 and considered by the examiner for awarding F grade in the course. Detection of ANY malpractice related to the lab course can lead to awarding an F grade in the course.

Naming Conventions for Submission

- Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz). The name of this file must be

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip`

  (Example: $ASSG1\_BxxyyyyCS\_LAXMAN.zip$). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.
- The source codes must be named as

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: $ASSG1\_BxxyyyyCS\_LAXMAN\_1.c$). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of 0 marks for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

- Violation 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_new.pdf.

QUESTIONS

1. 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:

   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

   Sample Output:

   0 1 4
   1 2
   2
   3 2
   4 3

2. 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 1:

```
a
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
```

Sample Output 1:

```
99
```

Sample Input 2:

```
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
```

Sample Output 2:

```
99
```

3. Write a program that implements Dijkstra's algorithm for computing shortest paths in a directed graph with positive edge weights. Assume that the nodes are labeled from 0 to $n-1$

Input Format:

- The first line of the input contains a positive integer $n \in [1, 1000]$, the number of nodes 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. If a node has no adjacent nodes, then the line corresponding to its adjacency list will contain the label of that node only.
- 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 positive real numbers in the range (0, 10000]. If a node has no adjacent nodes, then the line corresponding to its adjacent edge weights will contain the label of that node only.
- The rest of the input consists of multiple lines, each one containing a four-letter string followed by zero, one or two integers. The integers, if given, will be in the range 0 to n-1.
  - The string "apsp" is followed by a single integer, the label of the source vertex. Print the shortest path distance from the source vertex to all the n vertices in the graph, sorted in the order of their labels, in a space separated format. Print "INF" for nodes that are unreachable from the source vertex.
  - The string "sssp" is followed by two integers, respectively, labels of the source and destination nodes. Print the shortest path from the source node to the destination node, if such a path exists. Print "UNREACHABLE", otherwise.
  - The string "stop" means terminate the program.

Output Format:

- The output, if any, of each command should be printed on a separate line.

Sample Input:

```
9
0 1 4
1 5
2 3
3 6
4
5 2 7 8
6 2
7 4
8 5 7
0 2 20
1 3
2 7
3 5
4
5 1 6 4
6 0
7 2
8 2 1
apsp 0
sssp 0 6
sssp 0 7
sssp 5 6
sssp 8 7
sssp 4 0
stop
```

Sample Output:

```
0 2 6 13 12 5 18 10 9
18
10
```

13
1
UNREACHABLE

4. 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.

   (a) Breadth First Search (BFS)

   (b) 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 edge.
   - The last line contains the source vertex.

   Output Format:

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

   Sample Input:

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

   Sample Output:

   0 1 2 3
   0 1 2 3

5. Write a C 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:

   - 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 contains three integers x, y, w denoting there is a directed edge from x to y having a weight w.

   Output Format:

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

   Note: - The vertices are labeled from 0 to n-1.
   Sample Input:

   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

Sample Output:

-1

6. Write a program to implement a FIBONACCI HEAP H and perform the operations *insertion*, *deletion*, *extract_minimum*, *decrease_key* and *union*. Your program should contain the following functions:

   - MAKEHEAP() - Creates and returns a new heap H containing no elements.
   - INSERT(H, x) – Inserts a new node with key 'x' into the heap H.
   - MINIMUM(H) – Returns a pointer to the node in heap H whose key is minimum.
   - EXTRACTMIN(H) – Deletes the node with minimum key value from heap H.
   - DECREASEKEY(H, x, k) – **Decreases the value of node 'x' of the heap H to 'k'**, if node x's key is at least 'k'. Otherwise, it returns NIL.
   - DELETE(H, x) - Deletes the node with key 'x' from the heap H. (If node is not present, it returns NIL)

   Input Format:

   - Each line contains a character 'i', 'm', 'x', 'r', 'd' or 'e' followed by at most one integer. The integers, if given, are in the range $[-10^6, 10^6]$.
   - Character 'i' is followed by an integer separated by space; a node with this integer as key is created and inserted into H.
   - Character *'m'* is to print the node with minimum key value from H.
   - Character *'x'* is to delete the node with minimum key value from the heap H and the deleted node's key is printed.
   - Character *'r'* is followed by two integers separated by a space; a node with the first integer as key is searched and **its key value is decreased to the second integer** and the updated node's key is printed.
   - Character 'd' is followed by an integer separated by space; the node with this integer as key is deleted from H and the deleted node's key is printed.
   - Character 'p' is to **print the root list of the heap** H in which each nodes are separated by a space.
   - Character *'e'* is to 'exit' from the program.

   Output Format:

   - The output (if any) of each command should be printed on a separate line.

   Sample Input:

   i 10
   i 20
   i 30
   i 40
   i 50
   p
   m
   x
   p
   r 50 15

i 5
p
e

Sample Output:

10 20 30 40 50
10
10
20
15
5 15 20

NOTE: In order to print the root list of a fibonacci heap, the root list is traversed in the **right direction** starting from the minimum node, where the root list is implemented as a circular doubly linked list.

7. 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, and compute the efficiency of the disjoint set data structure find operation by applying neither, either or both of the heuristics, by counting the total number of data accesses performed over the course of the program. Your program must support the following functions:

   - MAKESET(X) - creates a singleton set with element x.
   - FIND(X) - finds the representative of the set containing the element x.
   - UNION(X,Y) - merges the sets containing elements x and y into a single 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.
   - 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', 's'} followed by zero, one or two integers. 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. Output the value of find(x) if x is found, and -1, otherwise.
     - Call the function union(x,y) if the input line contains the character 'u' followed by space separated integers x and y . Print -1, without terminating, 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 's', print the number of data accesses performed by the find commands by each of the data structures over the course of the program and terminate.

   Output Format:

   - The output consists of multiple lines of space-separated columns. The columns correspond to the following disjoint-set data structures:
     a. with neither ranked union nor path compression applied.
     b. with only ranked union applied.
     c. with only path compression applied.

d. with both ranked union and path compression applied.

- Each line in the output contains the output of the corresponding line in the input, after applying to the respective data structures.
- The last line of the output contains the number of data accesses performed by the find commands by each of the data structures over the course of the program.

Sample 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
s
```

Sample Output

```
1
2
3
4
5
6
7
8
9
1 1 1 1
3 3 3 3
5 5 5 5
7 7 7 7
9 7 9 7
5 5 5 5
3 5 3 5
1 5 1 5
1 5 1 5
10
10 5 10 5
38 32 33 30
```