Indian Institute of Technology Mandi
August-November 2018 Semester

CS202 - Advanced Data Structures and Algorithms

Programming Assignment 4

Course Instructor : Aditya Nigam

TA : Daksh Thapar

30 October 2018

1 Instructions

- Plagiarism is strictly prohibited. In case of violation of the same zero marks will be rewarded for this assignment and strict action will be taken.
- You must **not use STL library** classes and functions.
- Submit a Makefile which compiles all your codes.
- Students using CodeBlocks or other compilers are requested to make sure their code runs perfectly on Linux as mentioned in each problem. Your evaluation will be on computers in PC lab.
- You are required to **use the template files provided** in this assignment. For every problem, these templates are only to be used.

2 Implementation

- You are given 6 .hpp files. Each of the problems have to be solved using these files only.
- For each problem there will be a separate main file but every problem will share the .hpp files.
- For each problem, the code should save the inherent graph as adjacency list in a file named 'adj_lst.txt' and adjacency matrix in a file named 'adj_mat.txt'. The code should be capable of also taking the input directly from the adjacenecy list and matrix rather than the specified format defined for that problem.

10 marks are reserved for proper implementation and will only be awarded when all the above points are implemented for each problem. **10 marks**

3 Problems

1. **Breadth First Search:** Given a matrix that is filled with O, G, and W where O represents open space, G represents guards and W represents walls in a Bank. Our goal is to enhance the security of all open areas by optimally allocating guards near to it. In order to do so, we first want to replace all of the Os in the matrix with their shortest distance from any guard, without being able to go through any walls. Also, replace the guards with 0 and walls with -1 in output matrix. **10 Marks**

   Examples: O : Open Space; G : Guard; W : Wall

   Input: Input needs to be read from a file named 'input1.in' in the order specified below. Where each element in a line will be separated by a space. The user themselves have to ascertain the size of the input as it can vary.

   ```
   O O O O G
   O W W O O
   O O O W O
   G W W W O
   O O O O G
   ```

   Output: The output needs to be written in a file named 'output1.out' in the given format only.

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

2. **Depth First Search:** Given an undirected graph G, write a program that can print all connected components line by line. **10 Marks**

   Input format: First line will take the no of nodes(V) and number of edges(E). Next E lines will contain two nodes per line which have an edge between them. The nodes are 0 indexed. The input and output both will be communicated via files named 'input2.in' and 'output2.out' respectively.

   Sample Input:

   ```
   7 7
   0 1
   0 2
   ```

1 2
2 3
4 5
6 5
3 1

Output:
Connected Component 1: 0 1 2 3
Connected Component 2: 4 5 6

3. **Single Source Shortest Path:** Given a dictionary, and two words start and target (both of same length). Find length of the smallest chain from start to target if it exists, such that adjacent words in the chain only differ by one character and each word in the chain is a valid word (i.e., it exists in the dictionary). It may be assumed that the target word exists in dictionary and length of all dictionary words is same for 1 particular input but will vary for different inputs. **10 Marks**

   Input: A file 'dictionary.dic' containing the words in the dictionary in each line. There can be any number of words.

   A sample dictionary.txt file is given below.
   POON
   PLEE
   SAME
   POIE
   PLEA
   PLIE
   POIN

   The user will take the start and target as input via prompt during the code run.

```
./a.out dictionary.dic
The start word is: TOON
The target word is: PLEA
```

   Output:
   The path is:
   TOON
   POON
   POIN

```
POIE
PLIE
PLEE
PLEA
The Length is: 7
```

4. **Minimum Spanning Tree:** Aakash is a student of engineering college. On a working day, he has n classes in classrooms 1 to n respectively. He wants to be free from college as early as possible. The rule is that he has to enter in class 1. These class connections are encoded in a graph. To get free from college, attending the nth class is mandatory. He can't go to any class at any time. Certain connections exist between some classes encoded in graphs. He has to follow only that paths. Help him to get free from college as soon as possible. **10 Marks**

   Input format: First line will contain number of classes(n) and connections(p) exist. Next p lines will contain any two classes and duration of the path between them.

   Sample Input:

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

   Output format: First line should print the path and second line should print the total time he has taken from entering in class 1 to getting free from the college.

   Output:

   ```
   The path is: 1-> 2-> 3-> 5
   Total Time is: 11
   ```