Alexandra University
Faculty of Computers and Data Science.
Final Project Spring 2020

جامعة الإسكندرية
كلية الحاسبات
وعلوم البيانات

Algorithms and Data Structures **02-24-00108**

Submission will be according to the Faculty Rules & Timetable

Student/s may be discussed online after submission according to the announced timetable

General Guidelines

- The final project is an opportunity for you to apply the data structures and algorithms we've discussed this semester
- This project is to be done in groups of **1 - 4**
- You have to implement the project only with Java language
- You are not supposed to use the Built in data structures (only arrays are allowed)
- You have to implement **All** of the four problems below using the suggested data structures
- If 2 or more copies are discovered both parties will be considered failed without an extra chance for passing the course.

Deliverables

1. The project code as a zipped file
2. The project must include a README file that provides instructions on project dependencies, how to setup or install the code, how to run the code, and other relevant information.
3. In addition to your code, you should deliver fully documented report as a PDF file which contains the following
    I. Cover paper which contains Names, ids of students in the team
    II. Abstract that illustrates the group members and their individual contributions to the project and division of the work.
    III. Table of content
    IV. Problem statement
    V. Algorithm you used in the form of pseudocode or flowchart with illustrating required assumptions you may need
    VI. Time analysis and memory analysis for your code (Execution time or memory usage) in terms of the given parameters of each problem
    VII. Sample Runs

Problem 1: Hashing

You are supposed to input a string which may consist of

i. Lower case alphabets (a-z)
ii. Upper case alphabets (A-Z)
iii. Spaces
iv. Numbers (0-9)
v. Special characters like **!, - ,…** etc.

You are supposed to find out which character occurs the maximum number of times and the number of its occurrence, in the given string. If two characters occur an equal number of times, you have to output the character with the lower ASCII code.

For example, if your string was: aaaaAAAA, your output would be: A 4, because "A" has lower ASCII value (ASCII:65) than "a" (ASCII: 97).

You are supposed to deal with any string whose maximum length can be 1000 characters.

You have to use **hashing and build your own hash table** in your algorithm

Input format:

The input will contain a string.

Output format:

You've to output two things which will be separated by a space:

i) The character which occurs the maximum number of times.

ii) The number of its occurrence.

Sample input
Faculty of Computers and Data Science
Sample output
Space 5

Problem 2: Binary Search Tree

Given a set of N elements that contain exactly X distinct elements (while the rest are repetitions), the redundancy ratio is defined as the ratio N/X.

You are supposed to write a program to input a set of integers, and keep track of the redundancy ratio. Once the redundancy ratio reaches or exceeds a specific value r, the program terminates and prints this ratio with a message indicating that your numbers have "many repetitions". If you want to force the program to terminate before the redundancy ratio reaches r, you can input -1 as your input integer. In this case, the program prints the current ratio with a message indicating that your numbers have "only few repetitions".

Input format:

First line contains the redundancy ratio r as a real number greater than 1.

Next lines contain integers one per line.

Output format:

Print the required answer.

Sample input 1	**Sample input 2**	**Sample input 3**
1.5	1.2	2.0
1	1	4
4	2	6
2	4	8
5	5	2
2	1	5
2		5
		2
		-1
Sample output 1	**Sample output 2**	**Sample output 3**
ratio=6/4=1.5	ratio=5/4=1.25	ratio=7/5=1.4
many repetitions	many repetitions	only few repetitions

Problem 3: Topological Sorting

There are many islands that are connected by one-way bridges, that is, if a bridge connects islands a and b, then you can only use the bridge to go from a to b but you cannot travel back by using the same. If you are on island a, then you select (uniformly and randomly) one of the islands that are directly reachable from a through the one-way bridge and move to that island. You are stuck on an island if you cannot move any further. It is guaranteed that if there is a directed path from one island to the second there is no path that leads from the second back to the first. In other words the formed graph is a Directed Acyclic Graph.
Find the island that you are most likely to get stuck on; that is the island that you can possibly reach with the maximum number of paths from all other islands.

Input format:

First line: Three integers n (the number of islands), m (the number of one-way bridges), and r (the index of the island you are initially on)

Next m lines: Two integers u_i and v_i representing a one-way bridge from island u_i to v_i.

Output format:

Print the index of the island that you are most likely to get stuck on. If there are multiple islands, then print them in the increasing order of indices (space separated values in a single line).

Sample input

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

Sample output

4

Problem 4: Minimum Spanning Trees

You are given an undirected connected graph consisting of *n* nodes. All the nodes have distinct numbers from *1* to *n*. There are no self-loops in the graph. Now, you are allowed to remove edges from the graph.

The **cost** of removing an edge is |node1−node2| where node1 and node2 are the vertex numbers of node1 and node2 respectively, and the edge being removed has node1 and node2 as its endpoints.

By performing the above-mentioned operation as many times as you want, you need to convert the given graph into a single tree by using minimum possible cost. In the output, print the minimum cost required to achieve the given task.

Input format:

First line will contain two integers *n* and *m* where *n* denotes the number of nodes and *m* denotes the number of edges in the graph respectively.

The next *m* lines contain two integers u_i and v_i denoting an undirected edge between the vertices u_i and v_i.

Output format:

Print the required total minimum cost of removing edges.

Sample input

5 5

1 2

2 3

3 4

4 5

1 3

Sample output

1

Good Luck

Prof. Dr. Amr El Masry

Dr. Mervat Mikhail