COR-IS1702 Computational Thinking

Term 1 2020/21

Paper Assignment

Release: 21 Oct 2020 (Wed, Week 10)

1.5 weeks to attempt

Due: 1 Nov 2020, 11pm (Sun)

Instructions:

- This paper assignment consists of 4 questions. Each question has several parts detailed as follows:

Question 1 – Recursion / Complexity (15 marks)	(a), (b)
Question 2 – Stacks / Queues / Complexity (30 marks)	(a), (b)
Question 3 – Trees / Complexity (30 marks)	(a), (b), (c)
Question 4 – Graphs (25 marks)	(a), (b) (i) (ii) (iii)

- You are to do your own research and attempt each of the questions individually. Please refer to the student handbook for SMU Code of Academic Integrity.

- Your answers to all the four questions must be typed out and saved in a document named as **<your_section>_<your_full_name>.pdf**. Your program to Q4 is to be saved as **<your_secton>_<your_full_name>.py**.

- Submit both documents mentioned above to the course eLearn merged section smu.sg/ct2020 Assignment portal by the deadline stated above.

- Late submission will be subject to a penalty of 10% per day, up to 2 days after the indicated deadline. That is,
    - Submission after 11pm, 1 Nov 2020 and before 11pm, 2 Nov 2020 (10% penalty)
    - Submission after 11pm, 2 Nov 2020 and before 11pm, 3 Nov 2020 (20% penalty)
    - Submission after 11pm, 3 Nov 2020 (0 mark)

Question 1

(a) Determine the time complexity of the function `binary_min(arr)` below, based on the number of comparisons `min1 < min2` in line 8.

```
1:   def binary_min(arr):
2:     if len(arr) == 1:
3:       return arr[0]
4:     else:
5:       mid = len(arr)//2
6:       min1 = binary_min(arr[0:mid])
7:       min2 = binary_min(arr[mid:len(arr)])
8:       if min1 < min2:
9:         return min1
10:     else:
11:       return min2
```

(b) What is the time complexity of `binary_min(arr)` in Big-O notation? Show your derivation steps clearly.

Question 2

(a) Discuss how a stack can be implemented using two queues. Provide the `push`, `pop`, `peek` and `count` algorithms in terms of the queue's `enqueue`, `dequeue`, `peek` and `count` operations. Determine the Big-O time complexity in each case.

(b) Are you able to implement a stack using just one queue? If yes, please provide the `pop`(only) algorithm to simulate the stack operations with just one queue. If no, explain.

Question 3

(a) Design a non-recursive algorithm `print_kth_level_leaves(root, k)` that will take in the root of a binary tree and a positive integer k and print out the values of all the leaf nodes at level k of the tree. Note that the root is at level 1 and its child nodes are at level 2 and so on.

As an illustration, for the given binary tree t in figure 1:

- `print_kth_level_leaves(t.root,3)` will print D and G
- `print_kth_level_leaves(t.root,4)` will print F
- `print_kth_level_leaves(t.root,k)` will print nothing for k = 1 and 2 since there is no leaf nodes at level 1 and 2.

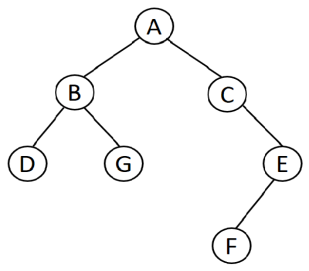

Figure 1. Binary tree t

(b) What abstract data type have you used in part (a). Justify how it has helped in your algorithm.

(c) What is the time complexity of your algorithm designed in part (a)? Explain your answer.

Question 4

(a) You aim to become a social network star and for that purpose you post funny videos of yourself. Your videos do not get many likes and does not become popular ☹. After some analysis you figure out that whether your video is popular or not depends on the total <u>number of friends and friends of friends</u> you have. You decide to find out this number by running BFS. Modify the BFS code in GraphLab.py so that you can count the number of your friends and friends of friends. Below are more requirements:

- The code should stop running after the counting is done, BFS should not run for the whole graph.
- You must submit python file(s) and also the code in a word/pdf doc - the word doc should highlight the lines that have been added or changed from GraphLab.py code. The code should be runnable and will be run by us when checking answers.
- Do not modify the Vertex or Graph object, all changes must only be in bfs_traversal or setVisitedBfs function only.

(b) Answer the following questions with explanation of less than 30 words, if required, for each part

(i) What is the out-degree of the right-most node in a topological sort?

(ii) Suppose you are given one of the topological ordering of a graph with <u>six nodes</u>. You add a back edge to the topological order, where a back edge is a directed edge from a node later in topological order sequence to an earlier node in the sequence. Provide an example graph (and the given topological ordering) where adding such a back edge does not result in a graph with cycles.

(iii) What is the maximum number of edges that a graph with n nodes can have when it is (i) undirected or (ii) directed? (note: we are not counting self-loops here, i.e., no edge from a node to itself. Also, in undirected graphs we count each undirected edge only once and not count them as two directed edges)