National University of Singapore

School of Computing

CS2040C - Data Structures and Algorithms

Final Assessment

(Semester 1 AY2022/23)

Time Allowed: 2 hours

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains THREE (3) sections.
   It comprises FOURTEEN (14) printed pages, including this page (page 14 is 'empty').
3. This is an **Open Book** but **not an Open Laptop** Assessment.
4. For Section A, use the OCR form provided (use 2B pencil).
   For Section B and C, answer **ALL** questions within the **boxed space** in the answer sheet.
   The answer sheet is at page 11-14.
   You will still need to hand over the entire paper as the MCQ section will not be archived.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some questions might be easier than they appear.
6. You can use **pseudo-code** in your answer only for Section C.
   You can use **standard, non-modified** algorithm discussed in class by just mentioning its name.
7. Please write your Student Number only. Do **not** write your name.

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Grading Remarks
A	39		
B	15		
C	46		
Total	100		

A MCQs ($13 \times 3 = 39$ marks)

Select the **best unique** answer for each question.
Each correct answer worth 3 marks.

The MCQ section will not be archived to open up possibilities of reuse in the future.

[PAGE 3 IS NOT ARCHIVED]

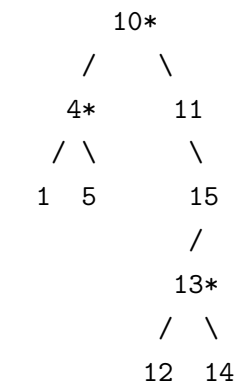
[PAGE 4 IS NOT ARCHIVED]

[PAGE 5 IS NOT ARCHIVED]

B Simpler Questions (15 marks)

B.1 BST Modification (6 marks)

You are given a BST that contains only **distinct** integer values. You want to output only the vertices in the given BST that have *exactly two* children in *descending* order. Please look at an example below. Vertices that have exactly two children are highlighted with a star (*). For this example, we output these integers: 13, 10, 4.



To standardize the answer, here are the relevant excerpts from BSTDemo.cpp show in class.
You just need to modify the `inorder` routine a bit (do that in the answer sheet and in C++).

```
struct BSTVertex { // all attributes are public to slightly simplify the code
  BSTVertex* left, right;
  int key;
};

template <typename T1>
class BST {
protected: // all other functions are hidden
  BSTVertex* root;

  void inorder(BSTVertex* T) {                  // modify this in the answer sheet
    if (T == NULL) return;
    inorder(T->left);                           // recursively go to the left
    cout << ' ' << T->key;                      // visit this BST node
    inorder(T->right);                          // recursively go to the right
  }

public:
  BST() { root = NULL; }

  void inorder() { inorder(root); cout << '\n'; }
};
```

B.2 IsTree (9 marks)

Please implement the following C++ 17 function `IsTree` that takes in an **undirected unweighted** graph stored in an Adjacency List and returns true if the graph is a tree, or return false otherwise. You will likely needs to call `dfs` routine and a sample implementation that uses std::vector `vis` to prevent cycle has been written for you (you have to use this version).

```
void dfs(int u, vector<vector<int>>& AL, vector<int>& vis) {
  vis[u] = 1;
  for (auto& v : AL[u])
    if (!vis[v])
      dfs(v, AL, vis);
}
```

The format of the Adjacency List `AL` is as discussed in class: An std::vector of V std::vectors of integers. std::vector u contains the indices of neighbors of vertex u. Note that bidirectional edge (u, v) will be stored *twice* in this format, i.e., edge $(u \rightarrow v)$ is stored in std::vector u whereas the other edge $(v \rightarrow u)$ is stored in std::vector v. Vertices are numbered between 0 to n-1.

A tree is as outlined in class: A connected (undirected) graph with V vertices and $E = V - 1$ (undirected) edges. There is only one unique path between any pair of vertices in the tree. All trees are also bipartite graphs.

```
bool IsTree(vector<vector<int>>& AL) {
  // implement this in the answer sheet (and in C++)
}
```

C Applications (14+15+17 = 46 marks)

C.1 Lexicographically Smallest Topological Sort (14 marks)

In class, we have learned about topological sorting of a Directed Acyclic Graph (DAG).

C.1.1 Count Valid Topological Sorts (4 marks)

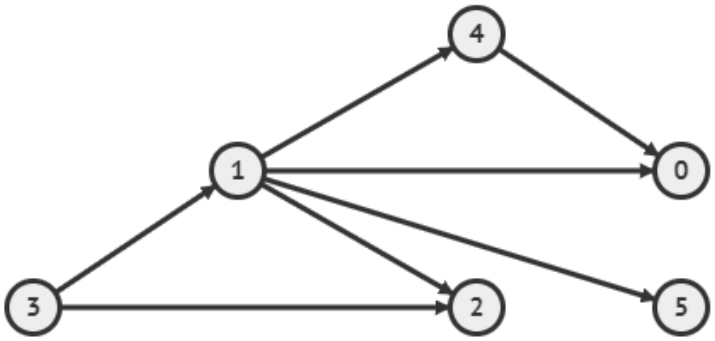


Figure 1: A DAG

Given a DAG as in Figure 1, *one of* the valid topological sort is $3 \to 1 \to 2 \to 4 \to 0 \to 5$.
This topological sort is also the lexicographically[1] smallest.
Your task is to count how many valid topological sorts are there in this DAG!
Please explain your counting method or enumerate *all* valid topological sorts of the DAG!

C.1.2 Get The Lexicographically Smallest Topological Sort (10 marks)

In class, we have learned two ways to get *any* valid topological sort: the DFS modification (postorder visitation) and the BFS modification (Kahn's algorithm). Neither can guarantee that the output is the lexicographically smallest toposort.

Your task is to show how to tweak the topological sorting algorithm further so that it can always output the *lexicographically smallest* topological order of any given DAG. To score up to 3 marks in this subsection, your correct algorithm should be $O(V! \times E)$ or $O(V!)$ — notice the factorial symbol. To score full (10) marks in this section, your correct algorithm should be $O((V + E) \log V)$ or better.

[1] The phrase lexicographic order means in alphabetical order.

C.2 String Similarities (15 marks)

Given two **short and equal-length** strings (each string has exactly k UPPERCASE characters), we define their similarities as the number of indices with the same character between the two strings. For example, the similarity of "SEVEN" and "SEVER" is 4 (the first four characters) whereas the similarity of "SEVEN" and "SAVER" is 3 (the first, third, and fourth characters).

Given n distinct strings of length k each, we define a special *tree* that connects all n strings with the *maximum* total similarities. There can be multiple possible trees as the solution, but our task is just to output this maximum total similarities. For example, if $n = 3$ strings of $k = 5$ characters: {"SEVEN", "SEVER", "SAVER"}, then it is best to connect "SEVEN" and "SEVER" (with similarity score of 4) and then connect "SEVER" (again) with "SAVER" (also with similarity score of 4), thus we output $4 + 4 = 8$ as the maximum total similarities of these $n = 3$ strings, as illustrated below.

C.2.1 One Manual Test Case (2 marks)

If you have understood this question, what is the answer (the maximum total similarities) if you are given $n = 4$ strings of $k = 3$ characters: {"CAT", "RAT", "BAT", "CAR"}? To convince the grader that your answer is not a random guess, show one possible special tree as shown in the example above.

C.2.2 What Is This Problem? (2 marks)

This problem is *similar* to one of the problem that we have discussed in class.
What is the name of this problem?

C.2.3 Solve This Problem (11 marks)

Propose an algorithm (and the associated data structure(s)) that is/are needed to solve this problem and analyze its time complexity. To score up to 8 marks in this subsection, your correct algorithm should be $O(n^2 \times (k + \log n))$. To score full (11) marks in this section, your correct algorithm should be $O(n^2 \times k)$ or better — Notice the disappearance of $O(\log n)$ in the time complexity. For avoidance of doubt, k is a small constant however you have to analyze your algorithm in terms of n and k.

C.3 Rock Climbing (17 marks)

Rock Climbing sport is getting more popular in Singapore these days. Your goal is to ring a bell that is hang exactly at H centimeters above the ground. You are a beginner of this sport and only has two possible moves that you have mastered. Move no 1: Jump up *exactly* U (integer) centimeters ($1 \leq U \leq H$). Move no 2: Drop down by *any* number of (integer) centimeters (using your safety harness – but obviously you cannot drop to lower than 0 centimeter, i.e., the ground level). To make the sport a bit more challenging, your gym coach has designated n disjoint forbidden zones (each zone ranges from height l to height r, inclusive, and $l \leq r$, also in (integer) centimeters) that you cannot touch during your ascent (or descent). It is guaranteed that the target height H is always not inside any forbidden zone. Your task is to compute the *minimum* number of moves that you need to reach the bell or output -1 if it is impossible to do so.

Example 1: $U = 40$, $H = 200$, $n = 1$ (from $l = 160$ to $r = 170$), then you cannot simply do 5 moves: 0 → (1. jump) 40 → (2. jump) 80 → (3. jump) 120 → (4. jump) 160* → (5. jump) 200 because height 160 is inside the forbidden zone [160..170] setup by your coach. The minimum answer is 8 moves (involving at least two drops), e.g., 0 → (1. jump) 40 → (2. jump) 80 → (3. jump) 120 → (4. drop down by 1 centimeter) 119 → (5. jump) 159 → (6. jump) 199 → (7. jump) 239 → (8. drop down by 39 centimeters) 200. PS: There are (many) other ways to solve Example 1 in 8 moves.

Example 2: $U = 40$, $H = 200$, $n = 1$ (from $l = 0$ to $r = 40$), then the answer is -1 as there is no way you can be off the ground without touching this very restrictive forbidden zone.

C.3.1 One Manual Test Case (2 marks)

If you have understood this question, what is the answer (the minimal number of moves) if you are given $U = 70$, $H = 290$, and $n = 2$ forbidden zones: [130..150] and [152..220]. To convince the grader that your answer is not a random guess, show one possible path as shown in the Example 1 above.

C.3.2 What Is This Problem? (2 marks)

This problem is a *variation* to one of the problem that we have discussed in class.
What is the name of this problem?

C.3.3 Solve This Problem (13 marks)

Propose an algorithm (and the associated data structure(s)) that is/are needed to solve this problem and analyze its time complexity. To score up to 8 marks in this subsection, your correct algorithm should be $O(H^2)$. To score full (13) marks in this section, your correct algorithm should be $O(H \log H)$ or better. Notice the major gap between the two time complexities and hence the 5 marks gap. For a full marks solution, it has to be able to solve test case with H up to 10^7 in 1s.

The Answer Sheet

My section B.1 answer:

```
void inorder(BSTVertex* T) {
```

}

My section B.2 answer:

```
bool IsTree(vector<vector<int>>& AL) {
```

}

My section C.1.1 answer (either explain or enumerate):

My section C.1.2 answer:

My section C.2.1 answer (draw the special tree):

My section C.2.2 answer:

My section C.2.3 answer:

My section C.3.1 answer (show the possible path):

My section C.3.2 answer:

My section C.3.3 answer:

This page 14 is for extra writing space if you need any.
But if you ever need it, Steven thinks you are *probably* already digressing to wrong answers...

– END OF PAPER; All the Best –