National University of Singapore

School of Computing

CS2040C - Data Structures and Algorithms
Final Assessment

(Semester 2 AY2018/19)

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.

3. This is an **Open Book Assessment**.

4. Answer **ALL** questions within the **boxed space** in this booklet.
   Only if you need more space, then you can use the empty page 14.
   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 but beware of penalty marks for **ambiguous answer**.
   You can use **standard, non-modified** algorithm discussed in class by just mentioning its name.

7. Please write your Student Number below. Do **NOT** write your name.

A	0	1						

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks
A	20		
B	30		
C	50		
Total	100		

A Basics (20 marks)

A.1 Worst Case Time Complexity Analysis (10 marks)

Write down the *tightest*[1] *worst case* time complexity of the various data structure operations or algorithms below. Each correct answer is worth 1 mark.

The operations (algorithms) referred below are the **unmodified version**, as per discussion in class, e.g. as currently explained in VisuAlgo or as currently implemented in C++ STL (gnu++17). Unless otherwise mentioned, there are currently n elements in the data structure. For graph-related operations, let n be the number of vertices and m the number of edges. You can assume that $10\,000 \le n \le 100\,000$. AM/AL/DAG/SSSP are the abbreviations for Adjacency Matrix/Adjacency List/Directed Acyclic Graph/Single-Source Shortest Paths, respectively. Unless specifically mentioned, all graph-related operations are performed on simple graphs stored in an AL data structure.

`l1` and `l2` are C++ STL `list` currently with n integers, `pq` is a C++ STL priority_queue currently with n integers, `ump` is a C++ STL unordered_map currently with n integers and bucket_count of m, `mp` is a C++ STL map currently with n integers. `v` is an integer. All integers mentioned in this section are 32-bit signed integers.

No	Operations		Time Complexities
1	`l1.splice(l1.end(), l2);`		$O(\underline{1})$
2	`for (int i = 0; i < n; ++i) cout << pq.top() << endl;`		$O(\underline{n})$
3	`auto pos = mp.upper_bound(v);`		$O(\underline{\log n})$
4	`for (auto &[k, v] :  ump) cout << k << ":  " << v << endl;`		$O(\underline{n})$
5	`for (auto &[k, v] :  mp) cout << k << ":  " << v << endl;`		$O(\underline{n \log n})$
6	Set all edge weights of an AM to 7		$O(\underline{n\text{\^{}}2})$
7	Find one of the topological ordering of a DAG		$O(\underline{n+m})$
8	Compute the number of connected component(s) of **a Tree**		$O(\underline{n})$
9	Compute the number of possible toposort(s) of **a Single Linked List**		$O(\underline{1})$
10	Compute SSSP from source s in an unweighted Graph		$O(\underline{n+m})$

---

[1]What we meant by tightest worst case time complexity is as follows: If an operation of the stipulated data structure/an algorithm needs at best $O(n^3)$ if given the worst possible input but you answer higher time complexities than that, e.g. $O(n^4)$ – which technically also upperbounds $O(n^3)$, your answer will be graded as wrong.

Section A.1 Marks = ____

A.2   Fill in the Blanks (10 marks)

Each correct answer is worth 1 mark.

1. You are given an array `A` that contains (just) $N = 2$ (32-bit) signed integers. What is the **best** algorithm to sort `A` in ascending order? ___just do min(n1, n2), and put the smaller number first___ _____.

2. You are given a non-circular Single Linked List `SLL` that contains $N = 100\,000$ (32-bit) signed integers. How many vertices in `SLL` have their `next` pointers not equal to `NULL`? ___99 999___.

3. You are given a Binary Max Heap Compact Array `A` that represents a Binary **Max** Heap `H` that contains $N = 100\,000$ (32-bit) signed integers. What is the best algorithm to find the **second smallest** integer in `H`? ___the best algo is to search all numbers in the compact array for the second smallest integer~~in the last level of the heap in O(n) time~~_____.

4. You are given a Hash Table `HT` with m cells. There are n distinct keys to be hashed into `HT`. We know for sure that there will be some collision(s) if $n > m$. However, if $n \le m$, is there a possibility that there is no collision? Please elaborate your answer. ___Yes there is a possibility. A scenario would be to hash for a table of 13 buckets 2 numbers like 41 and 42 with linear probing___.

5. You are given a BST `T` that has height (number of edges from root to the deepest leaf) $h = 7$ (not necessarily balanced). What is the **minimum** possible number of vertices in `T`? __7__ ~~O(n) algo***~~

6. You are given an AVL Tree `AvlT` that contains $N = 100\,000$ (32-bit) signed integers. What is the best way to retrieve the **second largest** integer in `AvlT`? ___in log n time find the largest integer, then find ~~the second largest integer by finding the~~ predeccessor___

7. You are given an unweighted Star Graph G_1 of size $V = 1\,000$ vertices (basically a tree with one internal (central) vertex – called vertex c – and V-1 leaves). If we store G_1 inside an Adjacency Matrix `AM`, describe how the content of `AM` will look like? ___vertex c's row and column will be completely filled_____.

8. You are given an unweighted general graph G_2 with $V = 10$ vertices labeled with $[0..V-1]$. When we run `dfs(2)` – Depth First Search (DFS) starting from source vertex 2 – on G_2 we notice that DFS visits the following set of vertices $\{0, 2, 7, 9\}$. If we now run `bfs(7)` – Breadth First Search (BFS) starting from source vertex 7 – on G_2 (all visited flags are re-set), which set of vertices that will be visited? _____.

9. You are given a Directed Acyclic Graph (DAG) G_3 with $V = 10$ vertices. Then you found out that G_3 has **V!** (that is, V-factorial) topological ordering of its V vertices. Therefore, G_3 must be a graph without ___a cycle_____.

10. You are given an undirected weighted graph G_4 with $V = 100\,000$ vertices and $M = 200\,000$ edges. The shortest path from vertex u to v in G_4 is just 1 unit. Therefore vertex u and v must be ___a complete graph_____.

Section A Marks = _____ + _____ = _____

B Intermediate (30 marks)

B.1 Analysis (9 marks)

Prove (show that the statement is correct) or disprove (give a counter example) the statements below.

1. Heap sort (as in C++ STL `partial_sort`) is a better sorting algorithm than randomized Quick Sort (as in C++ STL `sort`) in the following context: Sorting n 32-bit integers, n is up to 100 million integers, you have 4GB RAM in your computer, and its processor has 64KB/512KB/8MB of L1/L2/L3 cache memory, respectively.

   > Heap sort would take too much memory to store all 100 million integers into the heap, given 32 * 100 million bits is equal to 400 million bytes is more than 8 million bytes from 8MB, so MLE will be encountered.
   > Quicksort would not take that much memory since it uses the integers given to it and doesn't create additional memory, it only swaps elements in the array to sort using partition.

2. Suppose there is an array of integers A where the integer values are inside this rather large range $[0..1\,000\,000\,000]$. We can use a Hash Table instead of the standard integer frequency array in Counting Sort in order to sort A. This way, we avoid the Memory Limit Exceeded (MLE) issue as the range of integers inside A is big. Here is the snippet of the proposed change.

Before	After
`vector<int> f(1000000000, 0); // MLE`	`unordered_map<int, int> f;`
`for (int i = 0; i < n; ++i)`	`for (int i = 0; i < n; ++i)`
`  ++f[A[i]];`	`  ++f[A[i]];`
`for (int i = 0; i < 1000000000; ++i)`	`for (auto &[k, v] :  f)`
`  for (int j = 0; j < f[i]; ++j)`	`  for (int j = 0; j < v; ++j)`
`    cout << i << " ";`	`    cout << k << " ";`

3. As Binary Max Heap and Binary Search Tree (BST) properties are different, there can never be any Binary Tree of size $V \geq 2$ distinct integers that can be both Binary Max Heap and BST.

   > False. Given that the Heap is a complete tree with the MAX Heap property fulfilled, we can have a MAX Heap of 2 elements that can have a left child smaller than the root. The same can be said of a BST, where the left child is smaller than the root and its 2 elements. Ie: 5, 1 can be both a Max Heap and BST
   >
   > *** HOWEVER, if you're talking about a MIN Heap, then its not possible, as the MIN Heap left child is always bigger, while the BST left child is NEVER Bigger, but rather smaller

Section B.1 Marks = _____

B.2 Interesting Variants (12 marks)

You are probably familiar with the following standard declarations of C++ STL `stack`, `queue`, and `priority_queue` of integers:

```
stack<int> s;
queue<int> q;
priority_queue<int> pq;
```

All these three C++ STL classes are actually **wrappers of underlying containers**, which by default is a C++ STL `deque` for `stack` and `queue` or a C++ STL `vector` for `priority_queue`. But there is a way to ask `stack`, `queue`, or `priority_queue` to use a **specific** underlying container, e.g.

```
stack<int, deque<int>> s; // is 100% identical to stack<int> s
// stack uses back(), push_back(), and pop_back() of its container
queue<int, deque<int>> q; // is 100% identical to queue<int> q
// queue uses back(), front(), push_back(), and pop_front() of its container
priority_queue<int, vector<int>> pq; // is 100% identical to priority_queue<int> pq
// priority_queue uses front(), push_back(), pop_back(), and [] of its container
```

Knowing this, you are now interested to know if the following will work:

```
stack<int, vector<int>> s1;              // put your comment at box 1 below
stack<int, list<int>> s2;                // put your comment at box 2 below
queue<int, vector<int>> q1;              // put your comment at box 3 below
queue<int, list<int>> q2;                // put your comment at box 4 below
priority_queue<int, deque<int>> pq1;     // put your comment at box 5 below
priority_queue<int, list<int>> pq2;      // put your comment at box 6 below
```

Please **predict and elaborate** on what will happen to `s1`, `s2`, `q1`, `q2`, `pq1`, and `pq2` above.

The options are:

A. Cannot compile,

B. Becomes (slightly) slower,

C. Gives wrong behavior,

D. No significant change,

E. Becomes (slightly) faster.

Each box worth 2 marks.

1.

2.

3.

4.

5.

6.

Section B.2 Marks = _____

B.3   Create Test Cases (9 marks)

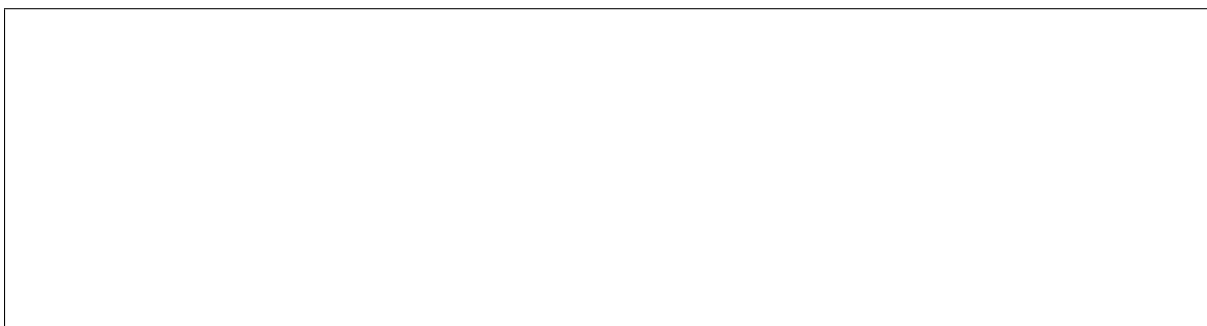Create Test Cases for each scenario below. Each valid test case worth 3 marks.

1. Draw a **connected** undirected graph with **exactly 7** vertices (labeled with [0..6]) and **at least 6** undirected edges such that both DFS(0) – Depth-First Search from source vertex 0 – and BFS(0) – Breadth-First Search from source vertex 0 – visit **exactly the same sequence** of 7 visited vertices.

2. Draw a Directed Acyclic Graph (DAG) with **exactly 5** vertices (labeled with [0..4]) and **exactly 4** directed edges such that there are **exactly 2** distinct topological ordering in this DAG.

3. Draw a directed weighted graph with **exactly 4** vertices (labeled with [0..3]) and **exactly 4** directed edges with **4 different edge weights** such that there are **exactly 2 shortest paths** from source vertex 0 to target vertex 3.

7

Section B Marks = _____ + _____ + _____ = _____

C Applications (50 marks)

C.1 Mayor of Porto 1879 (20 marks)

Let's time travel back to the year of 1879. You are the mayor of the city of Porto that year. Your city can be described as a 2D grid of $R \times C$ cells ($3 \leq R, C \leq 5\,000$). The grid contains either character '.' (body of water), character '#' (land, people can walk on foot), or character 'A' (that denotes a special place in Porto called Alfândega ("Customs" in Portuguese)). That year, the city of Porto is *not yet* connected due to the many body of water (Douro river, etc) that necessitates the usage of boat to connect two land cells that are currently unreachable without using a boat.

To make the city *more connected* (not necessarily fully connected) in the future (Porto hosted the ICPC World Finals in 2019), you – the city mayor that year – want to build k ($1 \leq k \leq 6$) bridges[2]. You have a bunch of good bridge engineers that can calculate what is the most efficient way to build a bridge over body of water to connect two previously disconnected land cells. Your job is to output X, the maximum number of land cells that are connected to Alfândega (inclusive) by choosing which k previously disjoint land *components* in Porto that you will connect with k new bridges.

For example, given the following 4×7 grid of a 'miniature' Porto (see the leftmost diagram), the output X is 2+5 = 7 land cells if you only have budget $k = 1$ (see the middle diagram) by building just one bridge between land component labeled with '4's (2 land cells, inclusive of Alfândega) and land component labeled with '1's (5 land cells). The output X is 2+5+3 = 10 land cells if you can build $k = 2$ bridges (see the rightmost diagram).

```
   .##....    |    .11....    |    .11....
   .###...    |    .111...    |    .111...
   .....##    |    ..|..22    |    ..|..22
   #.#A.#.    |    3.44.2.    |    3.44-2.
   -------------------------------------------------
   Original   |     k=1       |      k=2
```

Sample Input/Output 1 and 2

For the input, you will be given n, m, and k in one line, followed by n lines of m characters that describe the 2D grid. For the output, you need to just output X in one line.

```
   Input 1      | Output 1    | Input 2      | Output 2
   ------------------------------------------------------------
   4 7 1        | 7           | 4 7 2        | 10
   .##....      |             | .##....      |
   .###...      |             | .###...      |
   .....##      |             | .....##      |
   #.#A.#.      |             | #.#A.#.      |
```

---

[2]There are really 6 bridges that crosses over Douro river in Porto at the moment. One of the most famous one is Luís I Bridge built by Gustave Eiffel's team.

Please describe your solutions in **pseudo-code**. You will be graded on the following key points:

1. Are you going to store the graph of Porto? (5 marks)
   If yes, describe the details of your chosen graph data structure
   If no, describe on how you are going to traverse the graph

2. What (graph) algorithms that you will use to solve this problem? Please elaborate! (10 marks)

3. What is the time complexity of your solution overall? (5 marks)

Section C.1 Marks = _____

C.2 (Shortest) Directions to Alfândega, (20 marks)

You are given a map of the city of Porto as a directed weighted graph of n ($5\,000 \le n \le 10\,000$) vertices and m ($n - 1 \le m \le 200\,000$) edges. Each vertex/junction is labeled with integer [0..n-1]. The weight w of edge (u, v, w) describe the walking time w (in minutes) that one needs to walk along the road from junction u to junction v in Porto. As the city of Porto is hilly, walking downwards from higher junction to lower junction is usually faster than the other way around, i.e. weight w_1 of edge (u, v, w_1) may be different than weight w_2 of edge (v, u, w_2). As Porto is a modern city, it is connected, i.e. there will always be a way to go from any junction u to any other junction v in Porto.

You are the organizer of this competition and your job is to put directional signs on *every junction in Porto* (yes we are aware that it is overkill). These signs should point to the direction of the shortest path to Alfândega[3] (denoted as special vertex 7). If there are multiple such directions, you can just pick any one. The output should be n lines, output line $i \in [0..n - 1]$ is the vertex number j that has to be pointed by direction sign at vertex i so that someone is at vertex i, he/she knows that the shortest way to reach Alfândega is to go to vertex j (line 7 will point to nothing, i.e. '-1').

See the diagram below with $n = 8$ and $m = 17$ (only 4 edges are bidirectional with different weights). If someone is at vertex 6, then the shortest way to reach vertex 7 is not via direct edge $6 \to 7$ that has cost 8, but it is better to 'detour' via path $6 \to 2 \to 7$ that has cost $5+2 = 7$. Therefore, you should put direction sign at vertex 6 that points to vertex 2.
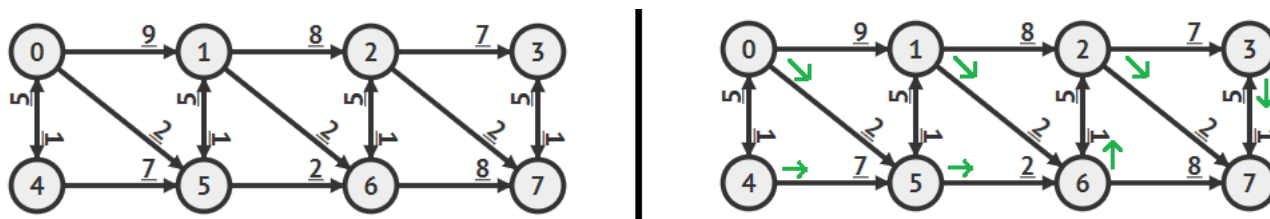

Figure 1: Left: The map of Porto, Right: One possible solution (Vertex 4 can also has its direction sign points to vertex 0 instead of vertex 5)

---

[3]The ICPC World Finals 2019 was conducted at this place.

Sample Input/Output

For the input, you will be given n and m in one line, followed by m lines of triple (u, v, w) that describe the m edges (0-based indexing).

For the output, you need to output n lines. For output line $i \in [0..n-1]$, print the vertex number pointed by vertex i.

```
    Input           | Output
    -------------------------------
    8 17            | 5
    0 1 9           | 6
    0 4 1           | 7
    0 5 2           | 7
    1 2 8           | 5
    1 5 1           | 6
    1 6 2           | 2
    2 3 7           | -1
    2 6 1           |
    2 7 2           |
    3 7 1           |
    4 0 5           |
    4 5 7           |
    5 1 5           |
    5 6 2           |
    6 2 5           |
    6 7 8           |
    7 3 5           |
```

Please describe your solutions in **pseudo-code**. You will be graded on the following key points:

1. Are you going to store the graph of Porto? (5 marks)
   If yes, describe the details of your chosen graph data structure
   If no, describe on how you are going to traverse the graph

2. What (graph) algorithms that you will use to solve this problem? Please elaborate! (10 marks)

3. What is the time complexity of your solution overall? (5 marks)

C.3 C++ Implementation of Either C.1 or C.2 (10 marks)

To get full marks for this paper, implement **either** C.1 or C.2 in **full C++ code**.

If you have enough time, you can attempt both and we will take the max score.

Note: Grading is very strict on this question, so only do this if you have completed other questions.

```cpp
#include <bits/stdc++.h>
using namespace std;
```

Section C Marks = _____ + _____ + _____ = _____

Extra blank paper:

– End of this Paper, All the Best –