National University of Singapore

School of Computing

CS2040C - Data Structures and Algorithms

Final Assessment

(Semester 2 AY2017/18)

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	25		
B	15		
C	60		
Total	100		

A Basics (25 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. 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. AM/AL/EL/BST/AVL/DAG are the abbreviations for Adjacency Matrix/Adjacency List/Edge List/Binary Search Tree/Adelson-Velskii Landis/Directed Acyclic Graph, respectively. Unless specifically mentioned, all graph-related operations are performed on simple graphs stored in an AL data structure.

No	Operations		Time Complexities
1	Reporting second largest integer in a C++ STL `priority_queue<int>`		$O($_____$)$
2	Inserting $n/2$ integers into a C++ STL `unordered_set<int>`		$O($_____$)$
3	Inserting $n/3$ integers into a not-necessarily-balanced BST		$O($_____$)$
4	Enumerate all values in an AVL tree that are $\in [x..y]$, $x \leq y$		$O($_____$)$
5	Modify weight w of an edge (w, u, v) inside an EL sorted by weight		$O($_____$)$
6	Add a new directed edge (u, v) into an AL		$O($_____$)$
7	Enumerate neighbors of a vertex u if we use an EL		$O($_____$)$
8	Test if two vertices u and v in a graph are *not* reachable		$O($_____$)$
9	Compute the shortest path from u to v in a weighted tree		$O($_____$)$
10	Compute the shortest path from u to v in an *unweighted* DAG		$O($_____$)$

[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. One of the best data structures to implement Stack ADT is _____.
2. Heap sort is basically the `extractMax()` operation of a Binary Max Heap that is called n times. It is already implemented in C++ STL algorithm routine called _____.
3. When the content of our hash table is frequently added and/or erased, it is better to use _____ collision resolution technique.
4. C++ STL `set` ignores duplicates. If we want to allow duplicates in our set, we shall use C++ STL _____ instead.
5. An AVL Tree with $n = 54$ values will have *at most* height $h =$ _____ (height of a tree is the number of edges from the root to the deepest leaf of that tree).
6. A tree is a special graph as it is connected, has $E = V$-1 edges, and has _____ between any pair of two vertices in the tree.
7. Starting from the same source vertex s, Depth-First Search will visit _____ set of vertices as Breadth-First Search.
8. To check if two vertices in a given graph are connected directly via an edge or indirectly via a path, we can use _____ algorithm.
9. The best algorithm to compute the shortest path from a source vertex to the other vertices is: _____. Assumption: We don't know if the graph has special structure but we are sure that the graph has no negative weight edge.
10. The optimized Bellman Ford's algorithm can be terminated earlier than the full V-1 rounds of all E edges relaxations if there is no _____ in the current round.

Section A.2 Marks = _____

A.3 Implications (5 marks)

Imagine that we have a working C++11 code that has a few (at least one) usage of C++ STL `set`. What are the implications (which can be either negative, *neutral*, or even *positive*) of adding 10 more characters '`unordered_`' so that all occurrences of C++ STL `set` in that C++ code are replaced with C++ STL `unordered_set`? Each correct answer is worth 1 mark.

1.

2.

3.

4.

5.

6. Optional:

Section A Marks = _____ + _____ + _____ = _____

B Analysis (15 marks)

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

1. Suppose that we implement a Hash Table that uses Separate Chaining collision resolution technique. We *can prove* that the load factor $\alpha = n/m$ where n is the number of keys and m is the Hash Table size can never be greater than a constant integer 7.

2. There is *no way* a **pre-order** traversal of a valid BST will yield a sorted (ascending) output.

3. Adjacency List is *always* a more efficient graph data structure in terms of memory usage compared to Adjacency Matrix.

4. There *can be more* than 2^n possible valid topological orderings in a DAG that has $n > 3$ vertices and m edges.

5. Bellman Ford's algorithm *may* produce wrong answer on some rare cases when solving the SSSP problem. This happens when there is a vertex that is unreachable from the source vertex s. You can assume that there is no negative weight cycle in the given graph.

Section B Marks = _____

C Applications (60 marks)

C.1 Powers of Two, 3 Digits Only (20 marks)

Disclaimer: This question is a simplified version of an UVa problem.

Steven wants to teach the concept of powers of two to his daughter Jane. He starts with a rather small positive integer x (not more than 3 digits) and tells Jane that basically $y = x^2$ is just $x \times x$. Steven then uses the value of y to be the next input value to be squared by Jane again, and so on.

However, as his daughter Jane is $\approx$ 7 years old (as of the date of this paper), she cannot do multiplication involving more than 3 digits *yet*. So if x^2 becomes greater than 3 digits, Steven will perform modulo 1 000 to the result before asking Jane to continue.

An example run if Steven starts from $x = 2$ is as follows: $2^2 = 4$, $4^2 = 16$, $16^2 = 256$, $256^2 = 65\,536$ (after modulo 1 000, we have 536), $536^2 = 287\,296$ (after modulo 1 000 we have 296), and so on.

Steven asks this question: How many different values of y (excluding the starting x) that he (and Jane) will see if Jane does all her calculations correctly?

Special question (**2 marks**): Is Steven able to continue generating different values of $y = x^2$ infinitely through this process? Why? _____.

Your solution (**13 marks)**: Design an efficient algorithm using the technique that we have learned in class (or beyond) to solve Steven's question above (the last **5 marks** for time complexity analysis):

// Extra writing space:

Let k be the answer produced by your algorithm. You can use variable k in your answer(s) below.

The time complexity of your algorithm is (**3 marks**): O(_____).

The space complexity required by your algorithm is (**2 marks**): O(_____).

Section C.1 Marks = _____

C.2 Road Blockages (20 marks)

Disclaimer: This question is a simplified version of a Kattis problem (For educational use only).

Every time there is an important person p from another foreign land arriving in an unnamed city, its government will block certain roads from members of the public. Fortunately, the blocked roads are *always* the roads along the shortest path from the International airport of that city (let's call it a) to the place (let's call it b) where p will make his/her public appearance. To reduce inconvenience, the government only blocks the affected roads from the moment p lands, i.e. time 0 until *exactly* 60 minutes later, i.e. time 60 (this is because the shortest path from a to b is guaranteed to be not more than 60 minutes).

You are not p. You are just an ordinary citizen. You want to go from your home (let's call it c) to your office (let's call it d) on a certain morning when the road blockages happen. You are quite smart, i.e. although a road is blocked, it does not mean that you cannot detour to reach your office. Note that such detour may not necessarily be the fastest way. There may also be a case that you need to traverse through at least one of the blocked roads in order to reach your office. In that case you have no choice but to accept that you will arrive at office 60 minutes later than usual that morning.

You are given:

1. The number of places in your city (positive integer V, not more than 10 000),
2. Four integers $\in$ [0..V$-$1]: a, b, c, d as described above, $a \neq b$, $c \neq d$,
3. The number of roads in your city (positive integer E, nor more than 100 000), and
4. The list of E roads in the city as integer tuples (u, v, w) that describes that there is a *directed* road connecting place u to v and can be traversed in w minutes, ($0 \leq u, v \leq V$-1; $0 \leq w \leq 1\,440$). It is guaranteed that there is a path from a to b and from c to d.

You task is to compute the minimum number of minutes to reach your office d, provided that the time you start your journey from c and the time p lands at a are exactly the same.

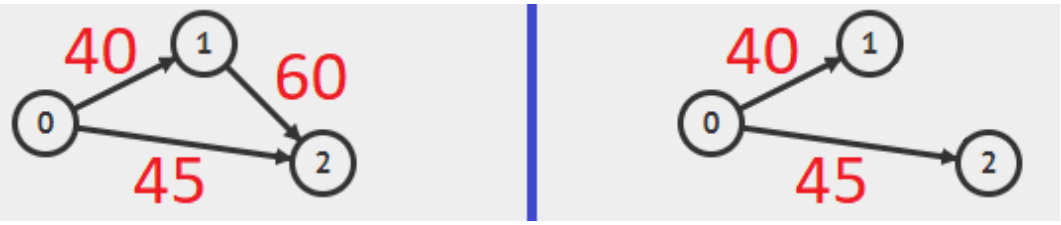

Figure 1: For both pictures, there are $V = 3$ vertices, $a = c = 0$, $b = d = 2$ (assume that you live next to the airport and work at the same place where p is headed); The shortest path from a to b is edge $a \rightarrow b$ that will be blocked for 60 minutes. On the right picture, we don't have edge $1 \rightarrow 2$.

Example 1a (Figure 1, left): For this test case, it is better for you to detour via path $0 \rightarrow 1 \rightarrow 2$ to arrive in $b = 2$ at time 40+60 = 100 instead of waiting for 60 minutes then traverse the previously blocked path $0 \rightarrow 2$ to arrive in $b = 2$ at time 60+45 = 105 (remember that you have to wait).

Example 1b (Figure 1, left): However, if the weight of edge $1 \rightarrow 2$ is > 65, then waiting to traverse edge $0 \rightarrow 2$ (that is blocked for the first 60 minutes) is actually faster.

Example 2 (Figure 1, right): Can't detour. Wait until edge $0 \rightarrow 2$ is available. The answer is 105.

Subtask (**6 marks**): Explain succinctly on how you are going to determine the roads that will actually be blocked for 60 minutes by the government, i.e. roads along the shortest paths from a to b:

The solution for the subtask above can be computed in $O($_____$)$ (**1 mark**).

The general case (**10 marks)**: Design an efficient algorithm using the technique that we have learned in class to solve the general case of this problem (the last **3 marks** for time complexity analysis):

// Extra writing space:

The time complexity of this general case algorithm is (**3 marks**): O(_____).

Section C.2 Marks = _____

C.3 Two Government Offices (20 marks)

Disclaimer: This question is modified from a Kattis problem (For educational use only).

There is a man who needs to clear a certain bureaucracy process involving two government offices.

There are n steps (numbered [0..n-1]) that the man has to complete and each step has to be done in either office no. 1 or in office no. 2. This information is given in a `vector` of n integers `which_office`, i.e. `which_office[u] = v` means that step `u` $\in$ [0..n-1] must be performed in office no. `v` $\in \{1, 2\}$.

There are m dependencies among those steps. This information is given in a `vector` of `vector` of integers `complete_before`, i.e. `complete_before[i] = vector-of-integers j` means that step `i` *must be completed* before the man can do *any* step mentioned in `vector` of integer `j`.

Traveling between the two offices is tiring. Thus, the man asks you what is the minimum number of trips that he has to make in order to complete all n steps. For this problem, assume that the man starts from office no. `1`. $1 \leq n \leq 100\,000; 1 \leq m \leq 1\,000\,000$.

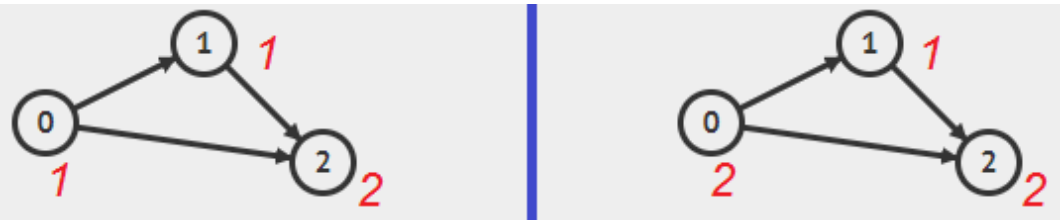

Figure 2: There are n vertices/circles/steps. The integer inside the circle is the step number `u`. The integer outside the circle is the value of `which_office[u]`. There are m edges/arrows. The edges describe the content of `complete_before`.

Example 1 (Figure 2, left): If `n = 3`, `which_office = {1, 1, 2}`, `m = 3`, `complete_before[0] = {1, 2}`, and `complete_before[1] = {2}`, then the answer is 1 as the man can start from office no. `1`, do both steps `0` and `1` there, then finally make **one single** trip to office no. `2` to complete the last step `2`.

Example 2 (Figure 2, right): If we have the same `n`, `m`, `complete_before` as in Example 1, but `which_office = {2, 1, 2}`, then the answer is 3 as the man has to first make **the first** trip from his starting office no. `1` to office no. `2`, do step `0` there, make **the second** trip back to office no. `1` to do step `1` there, and make **the third and final** trip back to office no. `2` to complete the last step `2`.

Special case 1 (**2 marks**): If all values in `vector which_office` are identical (all `1`s or all `2`s), then the answer is

The solution for the special case 1 above can be computed in O(_____) (**1 mark**).

Special case 2 (**2 marks**): If `m = 0`, i.e. there is no dependency requirement, but the n values in `vector which_office` are not identical like in special case 1 above, i.e. $\in \{1, 2\}$, then the answer is

The solution for the special case 2 above can be computed in $O($_____$)$ (**1 mark**).

The general case (**11 marks**): Design an efficient algorithm using the technique that we have learned in class to solve the general case of this problem (the last **3 marks** for time complexity analysis):

```
// compute the required answer
int count(int n, vector<int> &which_office, vector<vector<int>> &complete_before) {
```

// Extra writing space:

```
} // end of count function
```

The time complexity of this general case algorithm is (**3 marks**): O(_____).

Section C Marks = ____+____+____ = _____

Extra blank paper:

– End of this Paper, All the Best –