CS2040 - Data Structures and Algorithms

(Semester 1 AY2017/18)

Time Allowed: 2 Hours

Instructions

- Write your Student Number below, and on every odd page. Do not write your name.
- The assessment contains 6 multi-part problems. You have 120 minutes to earn 100 points.
- The assessment contains 22 pages, including this cover page and 5 pages of scratch paper.
- The assessment is closed book. You may bring one double-sided sheet of A4 paper to the assessment. You may not use a calculator, your mobile phone, or any other electronic device.
- Write your solutions in the space provided. If you need more space, please use the scratch paper.
- Show your work. Partial credit will be given. Please be neat.
- You may use any algorithm or data structure that we have studied in class (if unchanged), without restating it. If you want to modify the algorithm, you must explain exactly how your version works.
- Don't panic. Remember the general problem solving strategies we learned and apply them.
- Good luck!

Student Number.: ______________________

Problem #	Name	Possible Points	Achieved Points
1	True or False	20	
2	Queue with Max	15	
3	Leaves for Me	15	
4	h4x0r Alert	15	
5	Trading for Profit	15	
6	Connect Me Up, Scotty!	20	
Total:		100	

Problem 0. Before we begin. [0 points]

Circle the image that best represents how you feel right now.

😀	😆	😄	😎	🤔	😐	😴	😁	😠	😞

Problem 2. Queue with a Max [15 points]

In class, we saw how we could implement a Stack with a `max()` operation. One approach we discussed was to trade memory for speed and use two stacks: one to store the elements, and the other to store the max up to that element. What if we had to augment a Queue instead of a Stack?

Problem 2.a. [12 points] **Describe how you would implement a Queue with a `max()`.** For this problem, assume memory is cheap so, optimize for speed. **Be precise, but pseudocode/Java is not necessary unless it helps you explain.**

Your Answer:

Problem 2.b. [3 points] State the worst-case running times of the `enqueue`, `dequeue` and `max` operations of your solution. Provide a *brief explanation* to justify your answer.

Time Complexity:

`enqueue:` $\boxed{}$ `dequeue:` $\boxed{}$ `max:` $\boxed{}$

Brief Explanation:

Problem 3. Leaves For Me [15 points]

In many applications (like our Question Tree in Problem Set 2), the elements of interest are stored as leaves. As such, it is often necessary to get a list of the leaves in some order. This problem asks you to solve this problem for two different data structures.

Problem 3.a. [7 points] **Describe an efficient algorithm that prints out the leaves in a Binary Heap in *descending order*.**

State the time complexity of your algorithm for a Binary Heap with n elements. Assume that the heap is implemented using an array. **Be precise, but pseudocode/Java is not necessary unless it helps you explain.**

Time:

Your Algorithm:

Problem 3.b. [8 points] **Describe the most efficient algorithm you can think of that prints out the leaves in an AVL tree *in descending order*.**

Note: *Provide pseudocode or Java code* and state the time complexity of your method for an AVL tree with n elements.

Time:

Your Algorithm (with Pseudocode / Java code):

Problem 4. h4x0r Alert [15 points]

The notorious hacker `h4x0r` has infiltrated one of the computers in our company! He has already established connections from the compromised machine C to *all* $n-1$ other computers on the network. But he has yet to establish *bi-directional* communication, i.e., he can send commands to the other computers, but he can't get any information back (yet). We have to identify the compromised machine C before `h4x0r` causes any more damage.

Naruto has generated a $n \times n$ adjacency matrix A, where $A[i,j] = 1$ if there is communication from computer $i \rightarrow j$ and 0 otherwise.

For the first part of this problem, assume that our network is *connected* and that all communication is *symmetric*: $A[i,j] = A[j,i]$, *except* when one of the computers is the compromised machine C. There is an outgoing connection from C to all other computers on the network, but no incoming connection to C. No other computer has *only* outgoing connections to all other computers.

Problem 4.a. [7 points] **Describe the most efficient algorithm you can think of to identify C using the adjacency matrix A.**

State the time complexity of your method. Be precise, but pseudocode/Java is not necessary unless it helps you explain.

Time:

Explanation:

Faulty Assumption. It turns out that our assumption of *symmetric* connections between all computers was wrong! Some of the legitimate communication on our network happens to be *asymmetric*, i.e., there is a flow of information from $i \to j$ but not necessarily from $j \to i$. Note that there is still an outgoing connection from C to all other computers on the network, but no connection from any other computer to C.

Problem 4.b. [8 points] **Propose modifications to your solution (if necessary) or describe a new efficient method for finding the compromised machine C.**

State the time complexity of your algorithm. Be precise, pseudocode/Java is not necessary unless it helps you explain.

Time:

Your Algorithm:

Problem 5. Trading for Profit [15 points]

Boss comes to you with a neat idea. He thinks that it is possible to make money in a risk-free way by trading commodities (e.g., goats, gold, golf balls).

More precisely, you are given exchange rates for n commodities in a matrix E. The matrix specifies how much you can trade one item for another: one unit of i can be traded for $E[i, j]$ units of j. Assume that E is fixed (exchange rates don't change). Also, it is possible to trade a given item for any other item, and trade a fractional amount of items (e.g., 0.5 goats for 10 golf balls).

Boss's idea is essentially *arbitrage*: start with a single item of x and through a series of trades, end up with more than one unit of x. For example, given the matrix E below, there exists an arbitrage opportunity if we started with a goat: trade the goat for 0.1 gold, then trade the 0.1 gold for $0.1 \times 250 = 25$ golf balls, then trade the 25 golf balls for $25 \times 0.05 = 1.25$ goats. Yay! We now have (fractionally) more goats!

	Goats	Gold	Golf balls
Goats	1	0.1	20
Gold	10	1	250
Golf balls	0.05	0.004	1

Describe the most efficient algorithm you can think to determine if an arbitrage opportunity exists given E. State the time complexity of your method.

Be precise, but pseudocode/Java is not necessary unless it helps you explain.

Time:

Your Algorithm:

(Space for Problem 5)

Problem 6. Connect Me Up, Scotty! [20 points]

In an effort to unite its many peoples, the United Planets Federation (UPF) has decided to connect up its n member planets where n is large. Each planet i can be connected to another planet j via a *bi-directional* transdimensional space conduit that allows for faster-than-light (FTL) travel.

However, not all conduits can be made equal; some allow for a greater number of spaceships to pass through. The largest conduits can support up to 100 ships at a time, and the smallest only allow for up to 6 ships. We denote the number of spaceships that the conduit from $i \rightarrow j$ can support as $S[i, j]$. The UPF only has sufficient resources to build $n - 1$ conduits and would like to *maximize* the number of ships that can travel via the conduit network.

Problem 6.a. [6 points] **Describe the most efficient algorithm you can think of to determine which conduits to build.**

State the time complexity of your method. Be precise, but pseudocode/Java is not necessary unless it helps you explain.

Time:

Your Algorithm:

Poltical Interference. It turns out some planets are more friendly to each other than others. For example, Klangons do not like Valcons, but simply love Rosens. In the most recent Federation meeting, k pairs of planets have requested for conduits to be built between their worlds, and the UPF has agreed to their requests.

You are tasked to determine the remaining $n - k - 1$ conduits that the UPF should build. For this problem, assume that the conduits between the k pairs of planets do not result in a cycle and that $k < n/2$.

Problem 6.b. [6 points] **Describe the most efficient algorithm you can think of to determine the remaining conduits to build.**

State the time complexity of your method. Be precise, but pseudocode/Java is not necessary unless it helps you explain.

Time:

Your Algorithm:

Aliens! While building the conduits, the Federation made a stunning discovery: there already exists a conduit network built by ancient aliens a millennia ago! We want to use this alien network, but have to first check that it is *robust* to failures.

Let us define a conduit network as *robust* if by removing *any one* conduit, the network remains connected, i.e., there exists a path from each planet to every other planet.

Problem 6.c. [8 points] **Describe the most efficient algorithm you can think of to determine if the conduit network is robust.**

State the time complexity of your method. Be precise, but pseudocode/Java is not necessary unless it helps you explain.

Time:

Your Algorithm:

(*Space for Problem 6.c.*)

Problem 7. After you finish. [0 points]

Circle the image that best represents how you feel right now.

😃	😆	😁	😎	🤔	😐	😴	😬	😠	☹️

Take a moment to take a deep breath and relax. Make sure your Student Number is filled in. If you have time, check your work again, and try to come up with more efficient solutions.

Scratch Paper

Scratch Paper

Scratch Paper

Scratch Paper

Scratch Paper

End of Paper