CSE 250 - Midterm Exam - Version 1

3/15/23 @ 1PM, Norton 190

Name: _____ Person #: _____

UBIT: _____ Seat #: _____

Academic Integrity

My signature on this cover sheet indicates that I agree to abide by the academic integrity policies of this course, the department, and university, and that this exam is my own work.

Signature: _____ Date: _____

Instructions

1. This exam contains ~~9 total pages~~ (including this cover sheet). Be sure you have all the pages before you begin.
2. Clearly write your name, UBIT name, person number, and seat number above. **Additionally, write your UBIT name at the top of every page now.**
3. You have 50 minutes to complete this exam. Show all work where appropriate, but keep your answers concise and to the point.
4. After completing the exam, sign the academic integrity statement above. Be prepared to present your UB card upon submission of the exam paper.
5. You must turn in all of your work. No part of this exam booklet may leave the classroom.

DO NOT WRITE BELOW

P1	P2	P3	P4	P5	P6	(P7)	Total
18	20	17	20	10	15	(5)	100

Part 1 - Bounds and Runtime Analysis [18 Points]

1. For the following summation determine the unqualified ***O***, **Ω**, and **Θ** bounds, if they exist. For any bounds that do not exist, give a reason why. [4 points]

$$f(n) = \sum_{i=1}^{n^4} \sum_{j=1}^{n} 42i$$

***O*(*n*^{9}), Ω(*n*^{9}), Θ(*n*^{9})**

2. For the following function determine the unqualified ***O***, **Ω**, and **Θ** bounds, if they exist. For any bounds that do not exist, give a reason why. [4 points]

$$f(n) = 4log(2^{n^3}) + n^2 - 16$$

***O*(*n*^{3}), Ω(*n*^{3}), Θ(*n*^{3})**

3. For the following function determine the unqualified ***O***, **Ω**, and **Θ** bounds, if they exist. For any bounds that do not exist, give a reason why. [4 points]

$$f(n) = \begin{cases} log(n) & \textbf{if } n \textbf{ is prime} \\ 16n & \textbf{if } n > 10 \textbf{ and is even} \\ 19nlog(n) & \textbf{otherwise} \end{cases}$$

***O*(*n log*(*n*)), Ω(*log*(*n*)), Θ bound does not exist, because tight *O* and tight Ω are not the same.**

4. For the following functions, prove that $f(n) \in O(g(n))$ by finding appropriate values for the constants *c* and n_0. Show all work. [6 points]

$$f(n) = 6n^3 + 14n - 2$$

$$g(n) = 2n^3$$

Consider the following inequalities:

$6\,n^3 \leq c_1 * 2\,n^3$ ← This is true if $c_1 = 3$ and $n_0 = 0$

$14\,n \leq c_2 * 2\,n^3$ ← This is true if $c_2 = 7$ and $n_0 = 0$

$-2 \leq c_3 * 2\,n^3$ ← This is true if $c_3 = 1$ and $n_0 = 0$

So if we set c to 11 and n_0 to 0, we have:
f(n) ≤ c * g(n) for all n > n_0, therefore f(n) is in *O*(g(n))

[1 point] for valid c and n_0 or for showing the limit test
[2 points] for correct setup (showing what it means to be in O)
[3 points] for valid work

Note there are many constants that satisfy this inequality

Part 2 - Data Structure Selection [20 Points]

Each of the following two questions describe a particular use-case or application with a need that could be satisfied by the data structures we have discussed in class so far. **For each question you must state 3 things:** the ADT from class that best fits the usage pattern described, a specific data structure implementing that ADT that best meets the performance needs, and in at most one sentence give a reason why that data structure is the best choice.

5. Alice is working on processing a bunch of her old photos by writing a program that automatically applies filters and touch-ups to batches of photos all at once. Through some experimentation, she has found that working in batches of size 100 has the best performance. What ADT can be used to store these batches of photos, and what data structure should she use to implement that ADT and why? [10 points]

   [3 points] ADT: Sequence or Seq (1 point for queue or buffer)
   [3 points] Data Structure: Array (1 point for ArrayBuffer)
   [4 points] Reason: We have fixed size of 100, so we don't need the ability to add/remove, and there are advantages to having memory be contiguous. Updates are O(1). No extra memory overhead for Array

6. Bob is a user of popular social media site TwitTok, and he wants to learn more about how many people his posts have reasonable potential to reach. To do this, he downloaded data about all of the sites users and who they follow (the site's privacy is awful). He then plans to search through this data to find out how many people on the site are at most 3 steps away from him (people who follow him are one step away, people who follow someone one step away from him are two steps away, etc). What ADT would help him organize the data in a searchable way, what data structure should he use to implement that ADT, and what makes that data structure the best choice? [10 points]

   [3 points] ADT: Graph (1 point if mentioning queue for BFS)
   [3 points] Data Structure: Adjacency List
   [4 points] Reason: The runtime to search through the graph is smaller than for edge lists and adjacency matrices since each vertex stores its incident edges

Part 3 - Stacks and Queues [17 Points]

For questions in this part, consider the following code:

```
val seq = new MysterySequence()
seq.addSomething("S")
seq.addSomething("P")
seq.addSomething("A")
seq.addSomething("C")
print(seq.removeSomething())
print(seq.removeSomething())
print(seq.removeSomething())
seq.addSomething("E")
print(seq.removeSomething())
seq.addSomething("N")
print(seq.removeSomething())
```

7. What is printed if `MysterySequence` is a queue, and `addSomething` and `removeSomething` are `enqueue` and `dequeue` respectively? What are the contents of the queue after this code is run? [5 points]

   [4 points] SPACE, [1 point] remaining contents "N"

8. What is printed if `MysterySequence` is a stack, and `addSomething` and `removeSomething` are `push` and `pop` respectively? What are the contents of the stack after this code is run? [5 points]

   [4 points] CAPEN, [1 point] remaining contents "S"

9. State the unqualified and amortized runtimes of `addSomething` and `removeSomething` for `ArrayBuffer` and `LinkedList` implementations of `MysterySequence`. [7 points]

	ArrayBuffer		**LinkedList**	
	Unqualified	**Amortized**	**Unqualified**	**Amortized**
`addSomething`	**Θ(n)**	**O(1)**	**Θ(1)**	**Θ(1)**
`removeSomething`	**Θ(1)**	**Θ(1)**	**Θ(1)**	**Θ(1)**

[1 point] per correct answer (max of 7)

Part 4 - Arrays and Linked Lists [20 Points]

For questions in this part, consider the following code, in which `array` is an `ArrayBuffer[String]`, and `list` is a `LinkedList[String]`:

```
array.insert(idx = x, elem = "foo")
array.insert(idx = array.length, elem = "bar")
list.insert(idx = list.length, elem = "baz")
```

10. Assume we don't know anything about the value of `x`. What is the unqualified worst-case runtime for the call inserting `"foo"` in the above code if we assume that the underlying array in our `ArrayBuffer` is not full? How does your answer change if we cannot assume that the underlying array is not full? [5 points]

    [2 points] *O*(n), [3 points] without the assumption it is still *O*(n)

11. What is the unqualified worst-case runtime for the call inserting `"bar"` in the above code if we assume that the underlying array in our `ArrayBuffer` is not full? How does your answer change if we cannot assume that the underlying array is not full? [5 points]

    [2 points] *O*(1), [3 points] without the assumption it becomes *O*(n)

12. What is the unqualified worst-case runtime for the call inserting `"baz"` in the above code if list is a singly linked list? [5 points]

    [5 points] *O*(n)

13. What is the unqualified worst-case runtime for the call inserting `"baz"` in the above code if list is a doubly linked list? [5 points]

    [5 points] *O*(1)

Part 5 - Short Answer [10 Points]

14. In one or two sentences, describe the difference between an ADT and a data structure. Then name 2 ADTs and 2 data structures we have described in class. [5 points]

[1 point] ADTs just describe what you can do with the data, the data structure is the actual implementation of those capabilities.
[2 points] ADTS: Seq, Buffer, Stack, Queue, Graph
[2 points] Data Structures: Array, ArrayBuffer, LinkedList, EdgeList, AdjList, AdjMatrix

15. Consider the following runtime function for a recursive algorithm: [5 points]

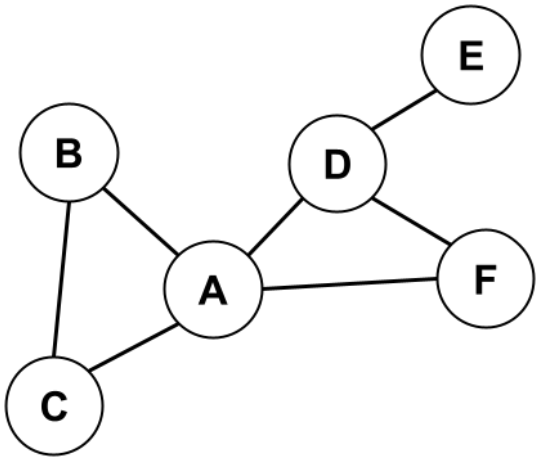
$$T(n) = \begin{cases} \Theta(1) & \textbf{if } n \leq 1 \\ 2 \cdot T(\frac{n}{2}) + \Theta(1) + \Theta(n) & \textbf{otherwise} \end{cases}$$

If we hypothesize that the runtime of this recursive algorithm is ***O*(*n* log(*n*))** state inequality we must prove for the base case, and state the inequality we must use as the assumption for the inductive case.

[2 points] Base: T(1) ≤ c * 1 * log(1)

[3 points] Assumption: T(n/2) ≤ c * n/2 * log(n/2)

Part 6 - Graphs [15 Points]



16. Draw below a spanning subtree of the above graph that could be found using a depth-first search starting at vertex **A**: [15 points]

[5 points] The drawn subtree must be a spanning subgraph (contain all nodes),
[5 points] a tree (no cycles),
[5 points] and for DFS ordering it must include edge BC and edge DF

Part 7 - Extra Credit [5 Points]

17. Is it possible to have a function, $f(n)$, that is in both $\boldsymbol{O(n^2)}$ and $\boldsymbol{\Omega(\log n)}$?
    If so, give an example.

    [1 point] Yes. [4 points] Many possible examples: n^2, log(n), n, case functions, etc.

Scrap Paper