CSE 247/502N Exam 1

Byeongchan Gwak

TOTAL POINTS

82 / 100

QUESTION 1

Ops and Algorithms 25 pts

1.1 Array access differences **2 / 2**

✓ **- 0 pts** Correct

1.2 Linked List access **2 / 2**

✓ **- 0 pts** Correct

1.3 LL Stack: pop() **2 / 2**

✓ **- 0 pts** Correct

1.4 LL Stack: push() **2 / 2**

✓ **- 0 pts** Correct

1.5 LL Stack: size() **2 / 2**

✓ **- 0 pts** Correct

1.6 Heap Height **1 / 1**

✓ **- 0 pts** Correct

1.7 Singly Linked List Ordered PQ add n **2 / 2**

✓ **- 0 pts** Correct

1.8 Singly Linked List Ordered PQ extract **2 / 2**

✓ **- 0 pts** Correct

1.9 Selection sort **2 / 2**

✓ **- 0 pts** Correct

1.10 Binary Heap insert() Omega() **0 / 2**

✓ **- 2 pts** Incorrect

1.11 Binary Heap insert() O() **2 / 2**

✓ **- 0 pts** Correct

1.12 Binary search of array O() **2 / 2**

✓ **- 0 pts** Correct

1.13 Linear search of Linked List O() **2 / 2**

✓ **- 0 pts** Correct

QUESTION 2

f(n) = Theta(n^3) 3 pts

2.1 f(n) = O(n^3) **1 / 1**

✓ **- 0 pts** Correct

2.2 f(n) = O(n^5) **1 / 1**

✓ **- 0 pts** Correct

2.3 f(n) = Omega(n^5) **1 / 1**

✓ **- 0 pts** Correct

QUESTION 3

Empirical and Analytical 14 pts

3.1 X vs. Y at n=10 **2 / 2**

✓ **- 0 pts** Correct

3.2 X vs. Y at n=100000 **0 / 2**

✓ **- 2 pts** Incorrect

3.3 P&Q **0 / 2**

✓ **- 2 pts** Incorrect

3.4 Java Lists remove from front **2 / 2**

✓ **- 0 pts** Correct

3.5 Java Lists remove from end **0 / 2**

✓ - 2 pts Incorrect

3.6 Java Lists get random index 2 / 2

✓ - 0 pts Correct

3.7 Java single add to ArrayList 0 / 2

✓ - 2 pts Incorrect

QUESTION 4

g(n) = O(n^3 log(n)) 3 pts

4.1 g(n) = O(n^3) 0 / 1

✓ - 1 pts Incorrect

4.2 g(n) = O(n^5) 1 / 1

✓ - 0 pts Correct

4.3 g(n) = Omega(n) 1 / 1

✓ - 0 pts Correct

QUESTION 5

Ticks 5 pts

5.1 Work 2 / 2

✓ - 0 pts Correct or mostly correct process

5.2 Result 3 / 3

✓ - 0 pts Correct

QUESTION 6

6 Prove 4n^2+3 is O(n^2 ln(n)) 4 / 5

✓ - 0 pts Correct

✓ - 1 pts Minor error or flaw / unconvincing work

1 top derivative incorrect

QUESTION 7

7 Min heap extractMin() 5 / 5

✓ - 0 pts Correct

QUESTION 8

8 Min heap insert(2) 5 / 5

✓ - 0 pts Correct

QUESTION 9

Recursive function 5 pts

9.1 a 2 / 2

✓ - 0 pts Correct

9.2 b 2 / 2

✓ - 0 pts Correct

9.3 f(n)'s Omega() 0 / 1

✓ - 1 pts Incorrect

QUESTION 10

Master method 9 pts

10.1 On any a and b? 1 / 1

✓ - 0 pts Correct

10.2 Show work 1 / 1

✓ - 0 pts Correct

10.3 Case 1 / 1

✓ - 0 pts Correct

10.4 Theta(...) 2 / 2

✓ - 0 pts Correct

10.5 Show work 1 / 1

✓ - 0 pts Correct

10.6 Case 1 / 1

✓ - 0 pts Correct

10.7 Theta(...) 2 / 2

✓ - 0 pts Correct

QUESTION 11

Costume Craziness 12 pts

11.1 PQs are Unordered Linked Lists, all lists are linked lists 2 / 3

✓ - 1 pts Error on $$n$$ term.

11.2 PQs are Unordered Linked Lists, allItems is ArrayList, other lists are linked lists **3 / 3**

✓ - 0 pts Correct

11.3 PQs are Heaps, allItems is ArrayList, other lists are linked lists **2 / 3**

✓ - 1 pts Error on $$b$$ and $$w$$ terms.

11.4 Part 1: in n **1 / 1**

✓ - 0 pts Correct

11.5 Part 2: in n **1 / 1**

✓ - 0 pts Correct

11.6 Part 3: in n **1 / 1**

✓ - 0 pts Correct

QUESTION 12

Recursion Trees 9 pts

12.1 Tree 1: a **1 / 1**

✓ - 0 pts Correct

12.2 Tree 1: b **0 / 2**

✓ - 2 pts Incorrect

12.3 Tree 1: f(n) **2 / 2**

✓ - 0 pts Correct

12.4 Draw tree **3 / 3**

✓ - 0 pts Correct

12.5 Tree height for n=256 **0 / 1**

✓ - 1 pts /4 at each level; $$log_4()$$

QUESTION 13

13 ID Number or Name **0 / 0**

✓ - 0 pts Correct

CSE 247 Fall 2021 Exam 1 Due at end of session

This exam is: **closed-book, NO electronic devices allowed,** and **closed-notes.** The exception is the "sage page" of the designated size on which you may have notes to consult during the exam.

Be sure you: **Provide legible answers in designated areas (credit will not be given for work that is difficult to read or not where expected); Clearly fill in circles (●) on multiple choice questions. Questions with circles require one choice; Leave the exam stapled together in its original order; Do *NOT* attach any other pages to the exam.** You are welcome to use the blank space on the exam for any scratch work.

If there are multiple "correct" answers for complexity, always pick the one that's the "best fit" (the lowest of the valid upper bounds or the highest of the valid lower bounds).;

If you need to leave the room for any reason prior to turning in your exam, you must leave your exam and any electronic devices with a proctor. **We do not clarify or explain anything during the exam session. State your assumptions if something is unclear and do the best you can.**

Question:	1	2	3	4	5	6	7	8	9	10	11	12	Total
Points:	25	3	14	3	5	5	5	5	5	9	12	9	100

You must complete all the identifying information below correctly. Failure to do so is grounds for a zero on this exam:

1. Name (**print** clearly): Byeongchan Gwak
2. Student ID (**print** clearly; 1 digit per underline): 5 0 1 0 2 6
3. You must sign the pledge below for your exam to count. The penalty for cheating will be decided during academic integrity review, but the instructors will recommend an F in this course as the minimum penalty.

I have read the instructions on this page and I will neither give nor receive any unauthorized aid on this exam.

Byeongchan Gwak

(Sign above)

⟹ ***Do not proceed until told to do so!*** ⟸

⟹ ***Initial the top right of each page before starting*** ⟸

1. (25 points) Data structure operations and common algorithms (true/false and multiple choice; See instructions on cover!):

   (1) Accessing the 1000th element of an array will take approximately 1000 times longer than accessing the first element.

   ○ True ● False

   (2) The time complexity of retrieving an element from a doubly linked list by index (like `x=list.get(1000)`) is considered to be $O(1)$.

   ○ True ● False

   (3) A singly linked list is used to create a `Stack` class. The list's `head` is used to keep track of the top of the stack and the `head` is the only instance variable in the class.

   i. The time complexity of a `pop()` operation is:

   ○ $O(\log(n))$ ● $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

   ii. The time complexity of a `push()` operation is:

   ○ $O(\log(n))$ ● $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

   iii. The time complexity of a `size()` operation is:

   ○ $O(\log(n))$ ○ $O(1)$ ● $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

   (4) A binary heap's height is:

   ● $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

   (5) A singly linked list is used for a *priority queue*. Items are stored in order by priority, with the highest priority item at the head of the list.

   i. The total time complexity of adding n new items is:

   ○ $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ● $O(n^2)$ ○ $O(n \cdot \log(n))$

   ii. The time complexity of extracting the highest priority item is:

   ○ $O(\log(n))$ ● $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

   (6) Lectures reviewed several variations on "selection sort". The best time complexity achieved was:

   ○ $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ● $O(n^2)$ ○ $O(n \cdot \log(n))$

   (7) Doing an `insert()` in a binary heap has an $\Omega()$ of:

   ● $\Omega(\log(n))$ ○ $\Omega(1)$ ○ $\Omega(n)$ ○ $\Omega(n^2)$ ○ $\Omega(n \cdot \log(n))$

   (8) Doing an `insert()` in a binary heap has an $O()$ of:

   ● $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

   (9) Binary search of data in an array has an $O()$ of:

   ● $O(\log(n))$ ○ $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

   (10) Linear search of data in a Linked List can have an $O()$ of:

   ○ $O(\log(n))$ ○ $O(1)$ ● $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

2. (3 points) Given that $f(n) = \Theta(n^3)$: $c_1 n^3 \le n \le c_2 n^3$

   (1) $f(n) = O(n^3)$

   ● True ○ False ○ Not enough information

   (2) $f(n) = O(n^5)$ $n \le c n^5$

   ● True ○ False ○ Not enough information

   (3) $f(n) = \Omega(n^5)$ $n \ge c n^5$

   ○ True ● False ○ Not enough information

3. (14 points) Empirical and Analytical Performance

   (1) Consider algorithms X and Y, which both sort data. When $n = 1000$ algorithm X takes 1 minute and algorithm Y takes 2 minutes.

   i. Algorithm X will be faster when $n = 10$: P 1000100

   ○ True ○ False ● Not enough information

   ii. Algorithm X will be faster when $n = 100000$: Q 2001500

   ● True ○ False ○ Not enough information

   (2) Consider algorithms P and Q. Algorithmic analysis shows that P will execute n^2+100 fundamental operations and Q will execute $2 \cdot n^2 + 1500$ fundamental operations. Consequently, P must be faster than Q when executed on any real computer with $n = 1000$:

   ● True ○ False

   (3) Consider the performance (time) of Java `List` implementations. Assume that k is large:

   i. You only care about removing k items from the *front*. Which is better:

   ● `LinkedList` ○ `ArrayList` ○ No clear difference

   ii. You only care about removing k items from the *end*. Which is better:

   ○ `LinkedList` ● `ArrayList` ○ No clear difference

   iii. You only care about *getting* k items from random indices. Which is better:

   ○ `LinkedList` ● `ArrayList` ○ No clear difference

   iv. A *single* `add()` to an `ArrayList` is:

   ○ $O(\log(n))$ ● $O(1)$ ○ $O(n)$ ○ $O(n^2)$ ○ $O(n \cdot \log(n))$

4. (3 points) Given that $g(n) = O(n^3 \cdot \log(n))$: $g(n) \le c \cdot n^3 \cdot \log n$

   (1) $g(n) = O(n^3)$ $g(n) \le c \cdot n^3$

   ○ True ● False ○ Not enough information

   (2) $g(n) = O(n^5)$ $g(n) \le c \cdot n^5$

   ● True ○ False ○ Not enough information

   (3) $g(n) = \Omega(n)$

   ○ True ○ False ● Not enough information

5. (5 points) Given the following pseudo-code, derive a precise, closed form equation for the resulting number of ticks in terms of n. Note: **1) Work must be shown where designated for credit and 2) the final result must be give on line below.**

```
i ← 1
while i ≤ n do
    tick
    tick
    j ← 0
    while j < i do
        tick
        tick
        j++
    end
    i++
end
tick
```

while

1. Show work:

inner $\sum_{j=0}^{i-1} 2 = 2(i-1+1) = 2i$

outer $\sum_{i=1}^{n}(2+2i) = 2n + 2\sum_{i=1}^{n} i = 2n + n(n+1)$

$= 2n + n^2 + n = n^2 + 3n$

And plus 1 tick at the bottom.

2. Give the final expression as a polynomial (in descending powers of n) below:

$ticks =$ $n^2 + 3n + 1$

6. (5 points) Big-O

Prove that $4n^2 + 3$ is $O(n^2 \cdot \ln(n))$ **Explain your work / steps:**

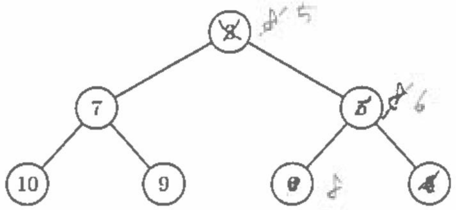
$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \implies f(n) = O(g(n))$

$\lim_{n\to\infty} \frac{4n^2+3}{n^2 \cdot \ln n} = \lim_{n\to\infty} \frac{16n}{2n \cdot \ln n + n^2 \cdot \frac{1}{n}} = \lim_{n\to\infty} \frac{16\cancel{n}}{2\cancel{n} \cdot \ln n + \cancel{n}}$

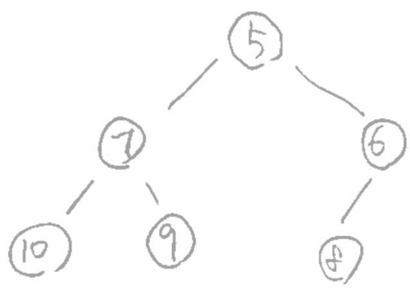
$= \lim_{n\to\infty} \frac{16}{2 \ln n} = 0$

$\therefore 4n^2+3$ has $O(n^2 \cdot \ln n)$

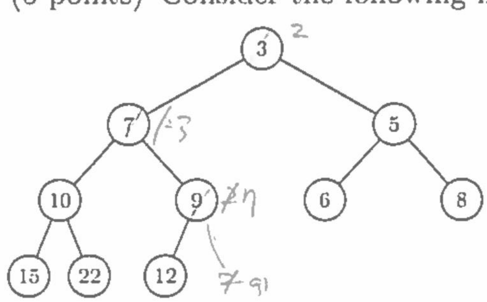
7. (5 points) Consider the following min-heap:



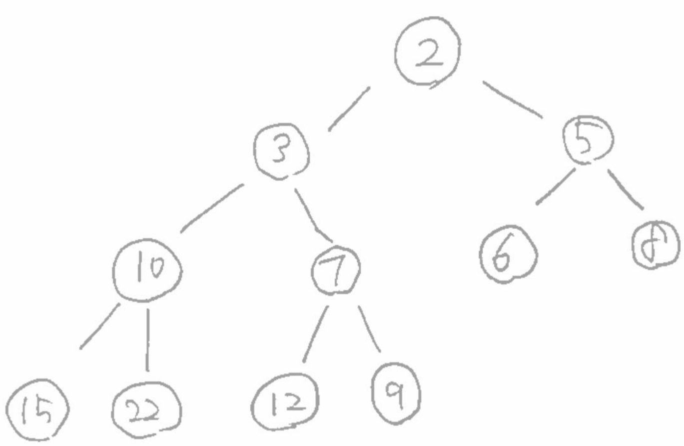
Show what it will look like following an `extractMin()`. Draw the complete, final heap below:



8. (5 points) Consider the following min-heap:



Show what it will look like following an `insert(2)`. Draw the complete, final heap below:



9. Given the general form of a recurrance: $T(n) = a \cdot T(\frac{n}{b}) + f(n)$

and the following algorithm:

Algorithm: reducto(array)
Input : array - a array of n items
if *array has 1 item* **then**

  `... /* <- This refers to other code that isn't shown */`

  return value

end

n=4

else

  `/* Make new arrays, each using parts of "array" */`
  first ← array$[0$ to $\lfloor \frac{n}{4} \rfloor]$
  second ← array$[\lceil \frac{n}{4} \rceil$ to $\lfloor \frac{2n}{4} \rfloor]$
  third ← array$[\lceil \frac{2n}{4} \rceil$ to $\lfloor \frac{3n}{4} \rfloor]$
  fourth ← array$[\lceil \frac{3n}{4} \rceil$ to $n-1]$

  `... /* <- This refers to other code that isn't shown */`

  return reducto(first) + reducto(second) + reducto(third)

end

The ... sections refer to code that isn't shown. They do basic data manipulation of the existing variables, but no recursive calls.

(1) (2 points) What is the value of a: 3

(2) (2 points) What is the value of b: 4

(3) (1 point) Based on the code that is provided, $f(n)$ is:

○ $\Omega(\log(n))$ ● $\Omega(1)$ ○ $\Omega(n)$ ○ $\Omega(n^2)$

10. Master Method

The basic version of the master method is:

$f(n) = O(n^c)$ where $c < c_{crit}$, then $T(n) = \Theta(n^{c_{crit}})$

$f(n) = \Theta(n^{c_{crit}} \log(n)^k)$ for $k \geq 0$, then $T(n) = \Theta(n^{c_{crit}} \log(n)^{k+1})$

$f(n) = \Omega(n^c)$ where $c > c_{crit}$ and $a \cdot f(\frac{n}{b}) \leq k \cdot f(n)$ for some $k < 1$ and large n, then $T(n) = \Theta(f(n))$

(1) (1 point) The master method can be used to solve recurrences for *any* values of a and b:

○ True ● False

(2) (4 points) Using the Master Method, what is the closed form for: $T(n) = 16 \cdot T(\frac{n}{2}) + n^4 \cdot \log(n)^2$

1. Show work/details

$a = 16 \quad b = 2 \quad C_{crit} = \log_2 2^4 = 4$ } $c = C_{crit}$.

$c = 4. (\because n^4)$

$\boxed{\text{case \#2}}$

$f(n) = n^4 \cdot \log(n)^2$

$= \Theta(n^{C_{crit}} \log(n)^k) \quad \therefore k = 2 \geq 0$

2. This is:

○ Case 1 ($c < c_{crit}$) ● Case 2 ($f(n) = \Theta(n^{c_{crit}} \log(n)^k)$) ○ Case 3 ($c > c_{crit}$)

3. Fill in the final expression for T(n) below:

$T(n) = \Theta(\underline{n^4 \cdot \log(n)^3})$

(3) (4 points) Using the Master Method, what is the closed form for: $T(n) = 4 \cdot T(\frac{n}{2}) + n \cdot \log(n)$

1. Show work/details

$a = 4 \quad b = 2 \quad C_{crit} = \log_2 2^2 = 2$ } $\Rightarrow c < C_{crit}$

$c = 1 \ (\because n)$

$\boxed{\text{case \#1}}$

$f(n) = n \cdot \log n$

$= O(n^c)$

2. This is:

● Case 1 ($c < c_{crit}$) ○ Case 2 ($f(n) = \Theta(n^{c_{crit}} \log(n)^k)$) ○ Case 3 ($c > c_{crit}$)

3. Fill in the final expression for T(n) below:

$T(n) = \Theta(\underline{n^2})$

11. Acme Costume Company has organized a global competition to judge Halloween costumes. Participants upload a photo of their costume, which is judged with an integer value (from 1 to 2,000,000,000. It's a big contest). Acme wants to give awards to both the *best* and the *worst* costumes. They've decided to utilize lists and priority queues to help them identify the winners. They are considering the following algorithm:

Algorithm: findWinners(allItems, best, worst, b, w)

Input : allItems - a list of entries; the judge's value is the priority

Input : best - an empty list. The *b* best costumes will be added when done

Input : worst - an empty list. The *w* worst costumes will be added when done

Input : b - The number of "bests" of interest

Input : w - The number of "worsts" of interest

```
n ← allItems.length
maxPQ ← new max priority queue
minPQ ← new min priority queue
for i ← 0 to n − 1 do
    item ← allItems.get(i)
    maxPQ.insert(item)
    minPQ.insert(item)
end
while b > 0 do
    item ← minPQ.extractMin()
    best.addLast(item)
    b ← b - 1
end
while w > 0 do
    item ← maxPQ.extractMax()
    worst.addLast(item)
    w ← w - 1
end
```

Continued on the next page...

$n(1+1+1) + b(n+1) + w(n+1)$

i) $3n + bn + wn$ $\quad n + bn + wn$

ii) $2n + bn + wn$ $\quad n + bn + wn$

$n(1+1) + b(n+1) + w(n+1)$

iii) $2n \cdot \lg n + b + w$ $\quad n\log n + b + w$

(1) Give the time complexity in terms of all three parameters, n, b, and w (that is, all three should be in the final equation), for each of the following conditions:

i. (3 points) If: a) both priority queues use unordered doubly linked lists, b) all three lists use doubly linked lists with head and tail references:

$O(n) =$ $n+bn+wn$

ii. (3 points) If: a) both priority queues use unordered doubly linked lists, b) `allItems` is an array-based list, but `best` and `worst` use doubly linked lists with head and tail references:

$O(n) =$ $n+bn+wn$

iii. (3 points) If: a) both priority queues use binary heaps, b) `allItems` is an array-based list, but `best` and `worst` use doubly linked lists with head and tail references:

$O(n) =$ $n\log n + b + w$

(2) If Acme decides to give many prizes, b and w may be approximately n. Give revised estimates of the above if b and w are approximately n (that is, only in terms of n):

n

i. (1 point) For the first variation:

$O(n) =$ n^2

ii. (1 point) For the second variation:

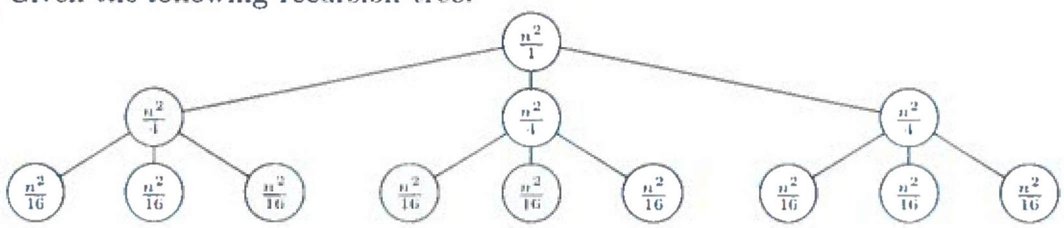
$O(n) =$ n^2

iii. (1 point) For the third variation:

$O(n) =$ $n \cdot \log n$

12. Recursion Trees

(1) Given the following recursion tree:

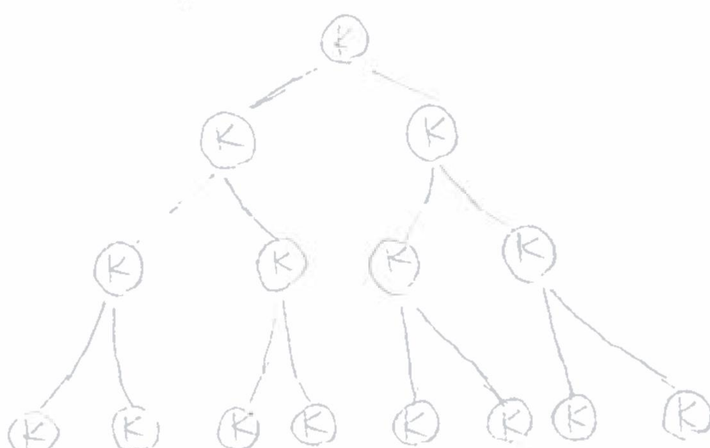


i. (1 point) What is a in the standard form of $T(n)$: 3

ii. (2 points) What is b in the standard form of $T(n)$: 4

iii. (2 points) What is $f(n)$ in the standard form of $T(n)$: n^2

(2) (3 points) Draw the recursion tree for $T(n) = 2 \cdot T(\frac{n}{4}) + k$ to a height of 4 (consider a root node alone to have a height of 1):



(3) (1 point) Using the previous recurrence (part 2), what height is needed for $n = 256$:

8

$256 = 2^8$