Q1 Big-O Justification

12 Points

Indicate whether the following assertions are true or false, and give a justification.

If you are justifying the positive direction, give choices of n0 and C. For big-Θ, make sure to justify both big-O and big-Ω, or big-O in both directions.

If you are justifying the negative direction, indicate which term(s) can't work because one is guaranteed to grow faster or slower than the other.

Q1.1 n + 5n^2 + 8n^4 is O(n^3)

2 Points

True [✓ False]

Justification

n^3 is a slower growing function than 8n^4. The correct staement would be n + 5n^2 + 8n^4 is O(n^4).

Q1.2 n! + n is O(n * log n)

2 Points

True [✓ False]

Justification

n * log n is a slower growing function than n!. The correct statement would be n! + n is O(n!)

Q1.3 log n + n * log n + log(log n) is Ω(n)

2 Points

[✓ True] False

Justification

Suppose n0 = 0 and C = 1

Q1.4 n^2 + n/4 + 6 is Θ(n^2)

2 Points

[✓ True] False

Justification

big-O justification: suppose n0 = 0 and C = 1/10
big-Ω justification: suppose n0 = 0 and C = 500

Q1.5 1/(n^50) + log32 is Ω(1)

2 Points

True [✓ False]

Justification

The constant function is a faster growing function than 1/(n^50) .
The correct statement would be 1/(n^50) + log32 is Ω(1/(n^51))

Q1.6 1/(n^50) + log2 is O(1)

2 Points

[✓ True]

False

Justification

Suppose x0 = 0 and C = 1000000

Q2 List Analysis

16 Points

Answer the following questions, and justify them with one or two sentences. Unless specified otherwise, all runtime is the worst-case runtime.

Format Instructions. Please read the following instructions carefully.

For your runtime, do not write Θ(), only what is inside of ().
Only give the most relevant terms. For example, if you arrived at the answer Θ(4n + 2), then this would simplify to Θ(n). Your answer would just be n.
For answers with exponential terms, you should format it using ^. For example, if your answer is $\Theta(n^2)$, your answer would be n^2.
For answers that involve log, you should explicitly write "log" with parentheses. For example, if your answer is Θ(log(n)), your answer would be log(n). You don't have to write the base of the log.
For answers with multiple terms, do not include spaces. For example, if your answer is Θ(n*m), your answer would be m*n. Additionally, order them alphabetically.

Q2.1 Did you read the format instructions?

0 Points

[✓ I have read and understood the format requirements]

Q2.2 Give a tight big-O bound for the best case running time of prepend in ArrayStringList

2 Points

Big O bound

1

Justification

The number of computations will always be the same. The best case is if there are no elements in the array and no need to expandCapacity(). The for loop runs a fixed number of times (constant).

Q2.3 Give a tight big-O bound for the worst case running time of prepend in ArrayStringList

2 Points

Big O bound

n

Justification

The worst case would occur if it would be necessary to expandCapacity() while the prepend method called the insert method. expandCapacity runs a for loop for n number of times in order to copy every single element into a new array.

Q2.4 Give a tight big-O bound for the best case running time of prepend in LinkedStringList

2 Points

Big O bound

1

Justification

If the String s was the first node, then this would be the best case. A new node would be created and added to the front. This is a constant function.

Q2.5 Give a tight big-O bound for the worst case running time of prepend in LinkedStringList
2 Points

Big O bound

1

Justification

Even if the linked list contained nodes before newFront, calling prepend would create a new Node, assign it to the front, and would point to the node that was previously in the front. This would only take one step (constant function).

Q2.6 Give a tight big-O bound for the best case running time of add in ArrayStringList
2 Points

Big O bound

1

Justification

The best case is if the array size does not have to be increased with expandCapacity(). In this case there are no for loops, and there would only be a fized amount of steps which makes it a constant function.

Q2.7 Give a tight big-O bound for the worst case running time of add in ArrayStringList
2 Points

Big O bound

n

Justification

The worst case is if there is a for loop to run that runs the method expandCapacity(). expandCapacity runs a for loop for n number of times in order to copy every single element into a new array.

Q2.8 Give a tight big-O bound for the best case running time of add in LinkedStringList
2 Points

Big O bound

n

Justification

Since this method has a while loop, the method has to run n number of times to iterate through the entire list of size n.

Q2.9 Give a tight big-O bound for the worst case running time of add in LinkedStringList
2 Points

Big O bound

n

Justification

Since this method has a while loop, the method has to run n number of times to iterate through the entire list of size n.

Q3 Mystery Functions and Measuring
26 Points

Determine which mystery method A-F corresponds to the implementations above by measuring against provided (but hidden) implementation.

Determine the tightest big-O bound for each function, and justify it with a few sentences. Give only the most relevant term, so use, for example *O(n)*, not *O(4n + 2)*.

Describe how you measured your functions.

Submit 3 relevant graphs that display your measurements.

Q3.1 Function Matching - Mystery A
1 Point

- f1
- [✓ f2]
- f3
- f4
- f5
- f6

Q3.2 Function Matching - Mystery B
1 Point

[✓ f1]

f2

f3

f4

f5

f6

Q3.3 Function Matching - Mystery C

1 Point

f1

f2

f3

f4

[✓ f5]

f6

Q3.4 Function Matching - Mystery D

1 Point

f1

f2

f3

[✓ f4]

f5

f6

Q3.5 Function Matching - Mystery E

1 Point

- f1
- f2
- f3
- f4
- f5
- [✓ f6]

Q3.6 Function Matching - Mystery F

1 Point

- f1
- f2
- [✓ f3]
- f4
- f5
- f6

Q3.7 Big-O bound for f1

2 Points

Big O bound

n

Justification

As seen by the graph for the mystery function B, the function illustrated is linear. Therefore, the runtime is n.

Q3.8 Big-O bound for f2

2 Points

Big O bound

n

Justification

As seen by the graph for the mystery function A, the function illustrated is linear. Therefore, the runtime is n.

Q3.9 Big-O bound for f3

2 Points

Big O bound

log(n)

Justification

As seen by the graph for the mystery function F, the function illustrated is logarithmic. Therefore, the runtime is log(n).

Q3.10 Big-O bound for f4

2 Points

Big O bound

n^2

Justification

As seen by the graph for the mystery function D, the function illustrated is a square function. Therefore, the runtime is n^2.

Q3.11 Big-O bound for f5

2 Points

Big O bound

n^3

Justification

As seen by the graph for the mystery function C, the function illustrated is a cube function. Therefore, the runtime is n^3.

Q3.12 Big-O bound for f6

2 Points

Big O bound

2^n

Justification

As seen by the graph for the mystery function E, the function illustrated is a exponential function. Therefore, the runtime is 2^n.

Q3.13 Description of measuring process

2 Points

The public class Measure had a method called measure which produced a list of measurements for a particular function. The arguments passed in were a String array of mystery function names, as well as a start and end number in order to run the mystery method. I first created a new list that would store the values of the measurements. Next, I looped through the passed in array if the length was greater than 1. After, I would iterate though the mystery function starting at the start value of n and ending at the end value of n (inclusive). I would also call the System.nanoTime() method at the start and end times. The result of the end-start times would be recorded as a new Measurement and added to the list.

Q3.14 Graphs

6 Points

A and B.pdf Download

1 / 1

C and D.pdf Download

E and F.pdf Download

1 / 1

Q4 Collaborators

0 Points

You are required to indicate whether or not you had collaborators on this assignment. If you did not have collaborators, type “I did not collaborate with anyone". Otherwise list the name of your collaborators.

I did not collaborate with anyone.

Programming Assignment 4 - questions - Late/Resubmit

● GRADED

STUDENT

Sophia Yermolenko

TOTAL POINTS

49 / 54 pts

QUESTION 1

Big-O Justification		**11** / 12 pts
1.1	n + 5n^2 + 8n^4 is O(n^3)	**2** / 2 pts
1.2	n! + n is O(n * log n)	**2** / 2 pts
1.3	log n + n * log n + log(log n) is Ω(n)	**2** / 2 pts
1.4	n^2 + n/4 + 6 is Θ(n^2)	**2** / 2 pts
1.5	1/(n^50) + log32 is Ω(1)	**1** / 2 pts
1.6	1/(n^50) + log2 is O(1)	**2** / 2 pts

QUESTION 2

List Analysis		**15** / 16 pts
2.1	Did you read the format instructions?	**0** / 0 pts
2.2	Give a tight big-O bound for the best case running time of prepend in ArrayStringList	**1** / 2 pts
2.3	Give a tight big-O bound for the worst case running time of prepend in ArrayStringList	**2** / 2 pts
2.4	Give a tight big-O bound for the best case running time of prepend in LinkedStringList	**2** / 2 pts
2.5	Give a tight big-O bound for the worst case running time of prepend in LinkedStringList	**2** / 2 pts
2.6	Give a tight big-O bound for the best case running time of add in ArrayStringList	**2** / 2 pts
2.7	Give a tight big-O bound for the worst case running time of add in ArrayStringList	**2** / 2 pts
2.8	Give a tight big-O bound for the best case running time of add in LinkedStringList	**2** / 2 pts

2.9	Give a tight big-O bound for the worst case running time of add in LinkedStringList	**2** / 2 pts

QUESTION 3

Mystery Functions and Measuring **23** / 26 pts

3.1	Function Matching - Mystery A	**1** / 1 pt
3.2	Function Matching - Mystery B	**1** / 1 pt
3.3	Function Matching - Mystery C	**1** / 1 pt
3.4	Function Matching - Mystery D	**1** / 1 pt
3.5	Function Matching - Mystery E	**1** / 1 pt
3.6	Function Matching - Mystery F	**1** / 1 pt
3.7	Big-O bound for f1	**2** / 2 pts
3.8	Big-O bound for f2	**2** / 2 pts
3.9	Big-O bound for f3	**2** / 2 pts
3.10	Big-O bound for f4	**2** / 2 pts
3.11	Big-O bound for f5	**2** / 2 pts
3.12	Big-O bound for f6	**2** / 2 pts
3.13	Description of measuring process	**2** / 2 pts
3.14	Graphs	**3** / 6 pts

QUESTION 4

Collaborators **0** / 0 pts