Name: Tuoxin Li Homework 4 - Analysis

Homework 4 - Analysis

In this homework, we are going to work to become comfortable with the mathematical notation used in algorithmic analysis.

Problem 1: Quantifiers

For each of the following, write an equivalent English statement. Then decide whether those statements are true if x and y are integers (e.g., they can be any integer). Then write a convincing argument to prove your claim.

1. $\forall x \, \exists y : x + y = 0$

There exists y for every x that x plus y equals to 0.

As x + y =0, then x = -y, which means that for every x there exists y which is equivalent to x with its negative value. As -y can represents the range of y, it’s proved.

2. $\exists y \, \forall x : x + y = x$

There exists y for every x that x plus y equals to x.

If x + y = x, then y = 0. When y is 0, then for every x that x + y = x is true.

3. $\exists x \, \forall y : x + y = x$

There exists x for every y that x + y =x.

This statement is false as for example, when y = 1, there can’t be x to satisfy the equivalence.

Problem 2: Growth of Functions

Organize the following functions into six (6) columns. Items in the same column should have the same asymptotic growth rates (they are big-Oh and big-θ of each other. If a column is to the left of another column, all of its growth rates should be slower than those of the column to its right.
n^2 , n!, $n\ log$2 n, 3n, 5n^2 + 3, 2^n, 10000, $n\ log$3 n, 100, 100n

Low					High
100	100n	nlog2n	n^2	2^n	n!
10000	3n	nlog3n	5n^2+3		

Problem 3: Function Growth Language

Match the following English explanations to the best corresponding big-Oh function by drawing a line from an element in the left column to an element in the right column.

Constant	$O(n$^3)
Logarithmic	O(1)
Linear	$O(n)$
Quadratic	$O(log$2 $n)$
Cubic	$O(n$^2)
Exponential	$O(n!)$
Factorial	O(2^2n)

Problem 4: Big-Oh

1. Using the definition of big-Oh, show that 100n + 5 ∈ $O(2n)$

The upper bound of 100n + 5 when n increases to a pretty large number is 100n(the constant is not important. Technically, we say 100n + 5 ∈ $O(n)$, and it’s fine to say 100n + 5 ∈ $O(2n)$ as we know it’s linear.

2. Using the definition of big-Oh, show that $n3 + n$^2 + n + 42 $\in O(n$^3)
When n increases to a pretty large number, n^3 will be the upper bound, so the Big-O of the formula is $O(n$^3).

3. Using the definition of big-Oh, show that n^42 + 1,000,000 $\in O(n$^42)
When n increases to a pretty large number, n^42 will be the upper bound, so the Big-O of the formula is $O(n$^42).

Problem 5: Searching

In this problem, we consider the problem of searching in ordered and unordered arrays:

1. We are given an algorithm called search that can tell us true or false in one step per search query if we have found our desired element in an unordered array of length 2048. How many steps does it takes in the worst possible case to search for a given element in the unordered array?

2048 steps.

2. Describe a fasterSearch algorithm to search for an element in an ordered array. In your explanation, include the time complexity using big-Oh notation and draw or otherwise clearly explain why this algorithm is able to run faster.

It should be binary search, and the time complexity is O(logn). Binary search is to divide the array into two parts with the middle element and find in which part we can find the data we want. We separate at maximum logn times to find the data we want.

3. How many steps does your fasterSearch algorithm (from the previous part) take to find an element in an ordered array of length 2,097,152 in the worst case? Show the math to support your claim.

Log(2, 2,097,152) ≈ 22

Problem 6: Another Search Analysis

Imagine it is your lucky day, and you are given 100 golden coins. Unfortunately, 99 of the gold coins are fake. The fake gold coins all weight 1 oz. but the real gold weighs 1.0000001 oz. You are also given one balancing scale that can precisely weight each of the two sides. If one side is heavier than the other the other side, you will see the scale tip.

1. Describe an algorithm for finding the real coin. You must also include the algorithm's time complexity. Hint: Think carefully – or do this experiment with a roommate and think about how many ways you can prune the maximum number of fake coins using your scale.

For the 100 coins, we separate them into 50 vs 50, weight, and choose the heavier 50. For the heavier 50, we separate them into 25 vs 25, weight, and choose the heavier 25. For the heaver 25, we separate them into 12 vs 12, weight, and if they are equally weighted, the left coin is real, otherwise, we choose the heavier 12. For the heavier 12, we separate them into 6 vs 6, weight, and choose the heavier 6. For the heavier 6, we separate them into 3 vs 3, weight, and choose the heavier 3. For the last three coins, we weigh 2 coins, and if they are weighted equally, the left one is real, otherwise, the heavier one is real.

2. How many weightings must you do to find the real coin given your algorithm?

Big-O is O(logn) and the worst case is that we separate logn times and finally find the real coin.

Thus, log(2,100) = 6.6. In this case, we must do 6 weightings.

Problem 7 – Insertion Sort

1. Explain what you think the worst case, big-Oh complexity and the best-case, big-Oh complexity of insertion sort is. Why do you think that?
The worst case is that the elements in the array are decreasing and big-o is O(n^2) because, for every outside for loop, there is an inside for loop to iterate and swap the elements.
The best case is that the array is already sorted. Big-O is O(n) in this case because only the outside for loop is iterating all the elements.
2. Do you think that you could have gotten a better big-Oh complexity if you had been able to use additional storage (i.e., your implementation was not in-place)?
No, because creating a new array or storing data in that array won't help.