Homework 7 - 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. $\forall x\, \exists y : x + y = 0$

For all x there exists a y such that x+y=0.

True.

Let y=-x,

x+y=x+-x=0

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

There exists a y for all x such that x+y=0.

True

Let y=0,

x+y=x+0=0

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

There exists a x for all y such that x+y=x.

False

If y is not equal to 0,

x+y is not equal to x.

Name: ____ Homework 7 - Analysis

Problem 2: Growth of Functions

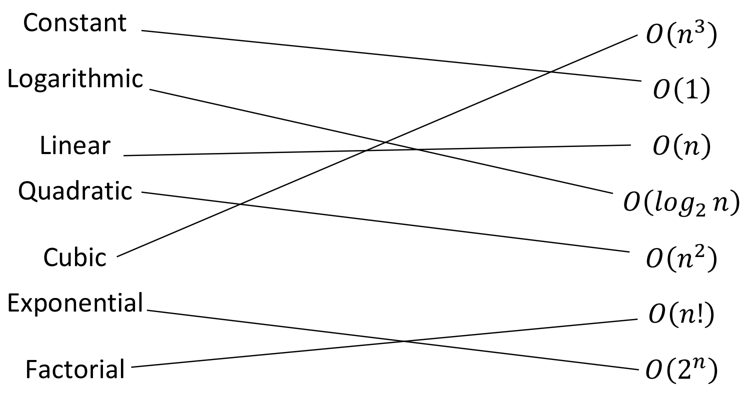
Organize the following functions into six (6) columns. Items in the same column should have the same asymptotic growth rates (they are big-Oh and big-θ of each other. If a column is to the left of another column, all of its growth rates should be slower than those of the column to its right.

$$n^2,\ n!, n\,log_2\,n, 3n, 5n^2+3, 2^n, 10000, n\,log_3\,n, 100, 100n$$

100	3n	n^2	$n\,log_3\,n$	2^n	$n!$
10000	100n	$5n^2+3$	$n\,log_2\,n$		

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.



Problem 4: Big-Oh

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

When n>=1,

100n+5<=100n+5n=105n

If c=52.5, when n>=1, $100n + 5 < \mathrm{c} * (2n)$

Therefore, $100n + 5 \in O(2n)$

2. Using the definition of big-Oh, show that $n^3 + n^2 + n + 42 \in O(n^3)$

When n>=1, $n^3 \geq n^2 \geq n \geq 1$

$n^3 + n^2 + n + 42$<=$n^3 + n^3 + n^3 + 42n^3$=45n^3

If c=45, when n>=1, $n^3 + n^2 + n + 42 \leq c * (n^3)$

Therefore, $n^3 + n^2 + n + 42 \in O(n^3)$

3. Using the definition of big-Oh, show that $n^{42} + 1{,}000{,}000 \in O(n^{42})$

When n>=1, $n^{42} \geq 1$

$n^{42} + 1{,}000{,}000 \leq n^{42} + 1{,}000{,}000n^{42} = 1{,}000{,}0001n^{42}$

If c=1,000,001, when n>=1, $n^{42} + 1{,}000{,}000 \leq c * (n^{42})$

Therefore, $n^{42} + 1{,}000{,}000 \in O(n^{42})$

Name: ____ Homework 7 - Analysis

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?

When we search for an element in an unsorted array, we need to search the elements one by one until we find the element we want. In worst case, we need to search all elements in the array and this takes 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.

We can use binary search to find an item in a sorted array. We find the middle item and compare it with the targeted item. If they are not equal, we continue to search in the half that may contain the targeted item and eliminate the other half. We repeat this process until we find the item. If all elements are eliminated, that means the targeted item is not in the array. It runs faster because it does not search the elements one by one. In each iteration it eliminates half of the remaining part, making it run less iterations than linear search. The big-O complexity of binary search is O(log(n)) while the big-O complexity of linear search is O(n). The running time of binary search increases more slowly than linear search.

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.

In worst case, using binary search to find an item in this array takes 21 steps. Since the big-O complexity of binary search is O(log(n)) and the size is 2097152, the result is log(2097152)=21

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.

Firstly, equally divide the coins into two sided. Each side has 50 coins. Since one side contains the real coin, it is heavier than the other side. So we keep the 50 coins from the heavier side and divide it into two sides again. Then we compare the weights of them and select the heavier side. After that, the side we keep have 25 coins and we are unable to divide it into two sides equally. In this situation, we randomly pick one coin out of the coins, which enables us to divide the 24 coins into two sides. If one side is heavier, we keep those 12 coins and divide it again. If the two sided are equal, that means the one we pick is the real one and we can stop. We will repeat this process until we ultimately find out the real coin.

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

In best case, we need 3 weightings. The first time we compare 50 coins from one side with 50 coins with another side. And we can only know the real coin is in one side. After that, we compare 25 coins from one side with 25 coins with another side. And again, we can only know the real coin is in one side. But in the third time, we pick one coin out of the 25 coins. If the weights of the two sides will be equal, the coin we pick is the real one. So in best case we must do 3 weightings.

In worst case, we need 6 weightings. 100/2=50. 50/2=25. 25/2=12. 12/2=6. 6/2=3. 3/2=1

Name: ____ Homework 7 - Analysis

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?

In best case, the big-O complexity of insertion sort is O(n). If the array is sorted, in each iteration it only compares two items, which takes one step, and then turn into next iteration. And the running time of iterations is equal to the size of the array. Therefore, the complexity of insertion sort is O(n).

In worst case, the big-O complexity of insertion sort is O(n^2). In each iteration it has to compares the first element in the unsorted subarray with all elements in the sorted subarray and moves all elements in the sorted subarray. And the running time of iterations is equal to the size of the array. So in worst case, the complexity of insertion sort is O(n).

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*)?

Yes. Merge sort requires additional storage and its complexity is O(nlog(n)).