Zengping Xu
CS 5008 2021 Spring
Homework Group: none

Problem 1 - Quantifiers

Write the following statements as English sentences, then decide whether those statements are true if x and y can be any integers. When deciding if x and y can be any integers, prove your claim with a convincing argument.

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

Claim: For every x there exists a y such that x + y = 0.
This is True since given any x, we have an existent y = -x, making x + y = 0.

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

Claim: There exists a y for all x such that x + y = x.
This is True since y = 0, for all x, x + 0 = x.

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

Claim: There exists a x for all y such that x + y = x.
This is False since when y = 5, x + 5 = x $\iff$ 5 = 0, statement false.

Problem 2 - Growth of Functions

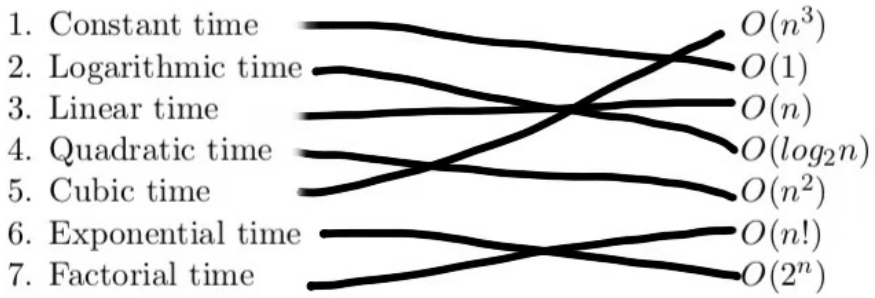
Organize the following functions into six columns. Items in the same column should have the same asymptotic growth rates (they are big-O and big-Θ of each other). If a column is to the left of another column, all its growth rates should be slower than those of the column to its right.

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

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

Problem 3 - Function Growth Language

Match the following English explanations to the best corresponding Big-O function by drawing a line from the left to the right.



Problem 4 - Big-O

1. Using the definition of big-O, show 100n + 5 = O(2n).

The particular C and n_0 are: C = 51, n_0 = 3

Claim: 100n + 5 is O(2n), then 100n + 5 ≤ C·2n, $\forall n > n_0$

Proof: Let C = 51, then we know

When $n_0 = 1$, the equation 100 + 5 ≤ 102 does not hold

When $n_0 = 2$, the equation 200 + 5 ≤ 204 does not hold

When $n_0 = 3$, the equation 300 + 5 ≤ 306 holds

When $n_0 = 4$, the equation 400 + 5 ≤ 408 holds

…

When $n_0 \to \infty$, 102n will always bigger than 100n + 5

So the equation 100n + 5 ≤ 102n holds for all n ≥ 3. By definition of Big O, 100n + 5 = O(2n) ☒

2. Using the definition of big-O, show $n^3 + n^2 + n + 100 = O(n^3)$.

The particular C and n_0 are: C = 10, n_0 = 3

Claim: $n^3 + n^2 + n + 100$ is $O(n^3)$, then $n^3 + n^2 + n + 100 \leq C \cdot n^3$, $\forall n > n_0$

Proof: Let C = 10, then we know

When $n_0 = 1$, the equation $1 + 1 + 1 + 100 \leq 10$ dose not hold

When $n_0 = 2$, the equation $8 + 4 + 2 + 100 \leq 80$ does not hold

When $n_0 = 3$, the equation $27 + 9 + 3 + 100 \leq 270$ holds

When $n_0 = 4$, the equation $64 + 16 + 4 + 100 \leq 640$ holds

…

When $n_0 \to \infty$, $10n^3$ will always bigger than $n^3 + n^2 + n + 100$

So the equation $n^3 + n^2 + n + 100 \leq 10n^3$ holds for all $n \geq 3$. By definition of Big O, $n^3 + n^2 + n + 100 = O(n^3)$ ⊠

3. Using the definition of big-O, show $n^{99} + 10000000 = O(n^{99})$.

The particular C and n_0 are: C = 2, n_0 = 2

Claim: $n^{99} + 10000000$ is $O(n^{99})$, then $n^{99} + 10000000 \leq C \cdot n^{99}$, $\forall n > n_0$

Proof: Let C = 2, then we know

When $n_0 = 1$, the equation $1 + 10000000 \leq 2$ dose not hold

When $n_0 = 2$, the equation $2^{99} + 10000000 \leq 2 \cdot 2^{99} \iff 1 + \frac{10000000}{2^{99}} \leq 2$ holds

When $n_0 = 3$, the equation $3^{99} + 10000000 \leq 2 \cdot 3^{99} \iff 1 + \frac{10000000}{3^{99}} \leq 2$ holds

…

When $n_0 \to \infty$, $2n^{99}$ will always bigger than $n^{99} + 10000000$

So the equation $n^{99} + 10000000 \leq 2n^{99}$ holds for all $n \geq 2$. By definition of Big O, $n^{99} + 10000000 = O(n^{99})$ ☒

Problem 4 - Searching

We will consider the problem of search in ordered and unordered arrays.

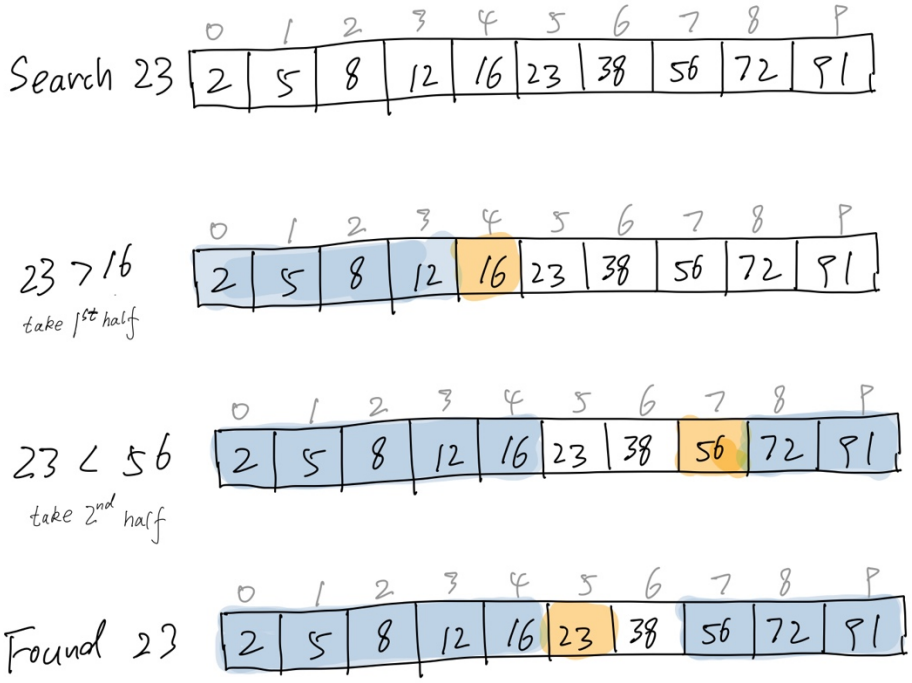
1. We are given an algorithm called search which can tell us true or false in one step per search query if we have found our desired element in an unordered array of length 2048. How many steps does it take in the worse possible case to search for a given element in the unordered array?

In worst-case we need 2048 steps to find the given element.

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

Using Binary Search: search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

The complexity is $O(\log_2 n)$. Since array has been ordered, using binary search we can avoid unnecessary compare by zooming off the searching range to half after every search.



3. How many steps does your fasterSearch algorithm (from the previous part) take to find an element in an ordered array of length 256 in the worse-case? Show the math to support your claim

According to the previous part, the complexity is $O(\log_2 n)$.

If the element is inclusive, in worst-case we need $\log_2 256 = 8$ steps to find an element.

If the element is exclusive, in worst-case we need $\log_2 256 + 1 = 9$ steps to find it.

We assume given element is the first element in an ascending array.

Step1: 256/2 = 128, compare k with array index 128, less than array[128]
Step2: 128/2 = 64, compare k with array index 64, less than array[64]
Step3: 64/2 = 32, compare k with array index 32, less than array[32]
Step4: 32/2 = 16, compare k with array index 16, less than array[16]
Step5: 16/2 = 8, compare k with array index 8, less than array[8]
Step6: 8/2 = 4, compare k with array index 4, less than array[4]
Step7: 4/2 = 2, compare k with array index 2, less than array[2]
Step8: 2/2 = 1, compare k with array index 1, less than array[1]

Then the element must locate in index 0, find it.

Problem 5 - 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 weigh 1 oz. but the 1 real gold weighs 1.0000000001 oz. You are also given one balancing scale that can precisely weigh each of the two sides. If one side is heavier than the other side, you will see the scale tip.

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

Algorithm: repeatedly tripartite dividing the coins. Begin with part1 : part2 : part3 = 33 : 33 : 34. Weighing the two parts with same coin number 33, if the scale tip shows one part is heavier, means real coin must be hidden in it. If the scale tip shows equal, then the real coin must be hidden in the third part with 34 coins. Tripartite dividing the target part coins in to11: 11 : 11 or 11: 11 : 12 and repeatedly check until the real coin is found.

The time complexity is $O(\log_3 n)$.

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

If the last weighing remaining 3 coins, then we must take $\text{Log}_3 100 \approx 4$ weighing.
If the last weighing remaining 4 coins left, then we must take $\text{Log}_3 100 + 1 \approx 5$ weighing.