CS 5008

Nianlong Lin

Assignment 11, CS5008 Fall2023

Problem 1 - 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, $100n$

Constant: 100, 10000

Linear: 3n, 100n

nlogn: $n\log_2 n$, $n\log_3 n$

Quadratic: n^2, $5n^2+3$

Exponential: 2^n Factorial: n!

slowest fastest

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

Problem 2 - 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.

1. Constant time → $O(1)$
2. Logarithmic time → $O(\log_2 n)$
3. Linear time → $O(n)$
4. Quadratic time → $O(n^2)$
5. Cubic time → $O(n^3)$
6. Exponential time → $O(2^n)$
7. Factorial time → $O(n!)$

Asymptotic Notation

Problem 3 - Big-O

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

formal definition: $f(x)$ is $O(g(x))$ if there exist constants c and k such that $|f(x)| \leq c|g(x)|$ whenever $x > k$

$100n + 5 \leq 100n + 5n$

$100n + 5 \leq 105n$

$C = 105 \quad k = 1/2$

let $C = 105$ $k = 1/2$,

$|100n + 5| \leq 105|2n|$, where $n > 1/2$.

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

$n^3 + n^2 + n + 100 \leq C \cdot (n^3)$

let choose $C = 2$, where $n > 10$

$10^3 + 10^2 + 10 + 100 \leq 2 \times 10^3$

$1210 \leq 2000$

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

$n^{99} + 10{,}000{,}000 \leq C \cdot n^{99}$

$\sqrt[99]{10{,}000{,}000} = 1.18$

let choose $C = 2$, where $n > 2$

$3^{99} + 10{,}000{,}000 \leq 2 \times 3^{99}$

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?

   since the array has 2048 elements, it would take 2048 steps in the worst case scenario to search for a given element in the unordered array.

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.

   this binary search has time complexity of $O(\log n)$. n is # of elements in the array.

   1) start from the middle of array.
   a) if target value is equal to the middle element, return the index
   b) if target value is greater, repeat the process with the right half of array.
   4) if the target value is smaller, repeat the process with the left half.
   5) if the array does not contain the target value, the search is unsuccessful.

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

   $\log_2 256 = 8$

   For an array of length 256, in the worst-case scenario, the fasterSearch would take 8 steps to find the element. This is significantly faster than a linear search, which would takes 256 steps in worst-case scenario.