Almazora, Thomas Jensen C. 5/17/24

CS121 – B54 LT - SW1

Constant Complexity

- This complexity will run in the same amount of time, no matter the input size.
- **Example**, if we want to get the first item in an array, no matter how large the array is, it will have the same time which for example, 50 milliseconds in all arrays.
- The notation of constant complexity is O(1).
- It is consistent in terms of efficiency, predictable performance, fundamentals operations which includes array indexing, variable access, and mathematical calculations that exhibit O(1) complexity and optimizing crucial code paths.
- It holds significance in algorithm analysis due to the runtime that remains constant regardless of the size of the input, creating predictable algorithms in situations where performance and responsiveness play a vital role.
- Constant space complexity is considered when the program doesn't contain any loop, recursive function, or call to any other functions.
- Another **example** is push and pop operation of fixed size stack, arithmetic operations, and comparisons.

Logarithmic Complexity

- This complexity refers to algorithm runtime increases very slowly compared to an increase in input size. The notation of logarithmic complexity is O(log N).
- Logarithms or log is a mathematical concept or expression that inverse (flip) of exponentials. It is the number of times that number multiplies itself to get a certain result. Typically answers the question, "How many times must one "base" number be multiplied by itself to get some other particular number or power to which one base number must be raised to produce another number.
- **Example**: How many times must a base of 30 be multiplied by itself to get 27,000? It is 3 (27,000 = 30 * 30 * 30).
- The most common logarithm is base-2. It frequently comes up when designing algorithms such as repeatedly dividing an array in half, doing bit operations, and many more.
- Binary Search , finding smallest or largest value in BST, and divide and conquer algorithms such as Merge sort and Quick sort are some of the examples that uses logarithmic complexity.
- The number of iterations of an algorithm will be represented by the exponent number when the complexity is logarithmical.
- It is designed to handle massive data sets with grace because of using base-2 and will always take half of the data out of the way.
- Another **example** is imagined you're in the library looking for one specific book among a hundred. The linear approach will scan every book one by one, while logarithmic approach will eliminate half the books with every step/iteration you take.

Linear Complexity

- This complexity refers to the runtime of the algorithm that grows almost linearly with the input size.
- **Example** would be looping over an array, the more elements there are in the array, the longer it takes to finish looping.
- It has a proportional relationship between the size of the input and the number of operations or steps required to solve the problem.
- The notation is O(n) which means that the function f(n) is proportional to n.
- Another **example** of linear algorithm is linear search wherein iterates through each element of an array until the target element is found. Array Traversal looped through each element of an array or list. Counting sort, which is a sorting algorithm wherein counts the occurrence of each element and reconstructs a sorted array. Finding maximum or minimum value from a linear array and searching for a value from an array.

Quadratic Complexity

- This complexity typically performs nested iteration, which means having a loop in a loop.
- **Example** is having an array with n items. The outer loop will run n times, and the inner loop will run n times also for each iteration of the outer loop, which will give total n^2 prints. If the array has 8 items, it will print 64 times.
- It is not very efficient and considered as bad space complexity but not the worst.
- The performance is directly proportional to the squared size of the input data set, which will occur whenever we nest over multiple iterations within the data sets.
- Another **example** is using bubble sort, insertion sort, selection sort, and traversing a matrix of 2D array.
- The notation of quadratic complexity is O(n^2).

Cubic Complexity

- This complexity runs in cubic time if the running time of the three loops is proportional to the cube of N.
- Typically occurs in using 3 loops. For **example**, matrix multiplication algorithm, multi-variable equation, and triplet nested loops run in cubic complexity.
- This time complexity is very slow and one of the worst time complexities.

Exponential Complexity

- This complexity follows a constant and n for the variable number. The notation for this complexity is O(2^n) which is constant power of the exponent number.
- **Example** of this kind of complexity is Brute Force algorithm. Another **example** is finding all unique combinations of something.
- It means that the running time of an algorithm doubles with each addition to the input data set.

- Denotes an algorithm whose growth doubles with each addition to the input data set. The time complexity starts off very shallow, rising at an ever-increasing rate until the end. Fibonacci numbers are one **example** using double recursive method.

References:

https://www.educative.io/courses/mastering-data-structures-and-sorting-algorithms-in-javascript/constant-complexity-o1

https://www.geeksforgeeks.org/what-does-constant-time-complexity-or-big-o1-mean/

https://www.freecodecamp.org/news/big-o-cheat-sheet-time-complexity-chart/

https://www.geeksforgeeks.org/constant-linear-space-complexity-in-algorithms/

https://www.geeksforgeeks.org/what-is-logarithmic-time-complexity/

https://towardsdatascience.com/logarithms-exponents-in-complexity-analysis-b8071979e847

https://www.baeldung.com/cs/logarithmic-time-complexity

https://javachallengers.com/logarithm-time-complexity/

https://paigeshin1991.medium.com/the-efficiency-of-logarithmic-time-complexity-c80746d4c0a1

https://www.educative.io/courses/mastering-data-structures-and-sorting-algorithms-in-javascript/linear-complexity-on

https://www.linkedin.com/advice/3/how-can-you-identify-linear-complexity-pattern-algorithm-sbltc

https://dev.to/luizcalaca/the-big-o-notation-quadratic-constant-and-linear-calculations-398a

https://www.linkedin.com/pulse/big-o-notation-time-complexity-algorithm-vikas-kumar/

https://www.youtube.com/watch?v=79QT5TIopcQ&ab_channel=NaveenAutomationLabs

https://javachallengers.com/big-o-notation-explanation/