CS301
Data Structure and Algorithms
Lecture 2: Time and Space complexity

Pandav Patel
Assistant Professor

Computer Engineering Department
Dharmsinh Desai University
Nadiad, Gujarat, India

Objective

- Understand what it takes to produce efficient computer program
- Understand time complexity of a program
- Understand space complexity of a program

What constitutes a computer program?

- Data + Algorithm = Program
- Choice of an algorithm is very important for a program
- Choice of structure to store data is equally important
- Choice of algorithms and data structures are dependent on each other
    - A task can be completed by different algorithms and all of them will run efficiently with specific data structures

How to measure efficiency of a computer program?

- Efficient program minimizes resource utilization
- Primary resources for a program are memory (space) and time
- How to analyse/measure space and time utilized by a program?
  - A priori estimates - Performance analysis
    - Does not require implementation
    - Provides estimates
    - Asymptotic notations - e.g. Big Oh
  - A posteriori testing - Performance measurement
    - Requires actual implementation
    - provides actual measurement
    - Tools like *gprof* could be used
    - Time required to run a program for the same input varies from one machine to the other

Time and Space complexity

- Time Complexity
  - Amount of time required to run to completion
  - Generally expressed in terms of input size
  - We will use big oh notation to express time complexity
- Space Complexity
  - Amount of memory (space) required to run to completion
  - Generally expressed in terms of input size
  - We will use big oh notation to express space complexity

Problem I

- What will be the space and time complexity of a search operation in an input of n elements?

Problem II

- What will be the space and time complexity of m search operation in an input of n elements?

Problem III

- What will be the space and time complexity of m search operation in an input of n sorted elements?

Problem IV

- What will be the space and time complexity of searching m sorted elements in an input of n sorted elements?

Problem V

- What will be the space and time complexity of sorting an input of n unsorted elements using selection sort algorithm?
- Are there any better algorithms for sorting in terms of time complexity?

Problem VI

- What will be the space and time complexity of addition of two $n \times n$ matrices?

Problem VII

- What will be the space and time complexity of multiplication of two n x n matrices?

Problem Solutions

- Problem I
  - If element to be searched is given before the list of elements
    - Time complexity - Linear - $O(n)$
    - Space complexity - Constant - $O(1)$
  - If element to be searched is given after the list of elements
    - Time complexity - Linear - $O(n)$
    - Space complexity - Linear - $O(n)$

Problem Solutions

- Problem II
  - Approach I: Linear search
    - Time complexity - $O(m \ x \ n)$
    - Space complexity - $O(n)$
  - Approach II: Binary search
    - Array to be sorted using selection sort
    - Time complexity - $O(n^2 + m \ x \ log(n))$
    - n^2 for selection sort and $m \ x \ log(n)$ for binary search of m elements in list of n elements
    - Space complexity - $O(n)$
    - Space and time complexity will vary based on choice of sorting algorithm
    - THINK: Is binary search possible on unsorted array? If yes, how? If no, why?
    - THINK: Is binary search possible on sorted linked list? If yes, how? If no, why?
  - Which approach is better?
    - Depends on values of n and m

Problem Solutions

- Problem III
  - Approach I: Linear search
    - Time complexity - $O(m \text{ x } n)$
    - Space complexity - $O(n)$
  - Approach II: Binary search
    - Input list is already sorted
    - Time complexity - $O(m \text{ x } log(n))$
    - One element can be searched in the list in logarithmic time - $O(log(n))$
    - Space complexity - $O(n)$
  - Which approach is better?
    - Binary search
  - Can there be any other approach? Will it help if we sort m elements?

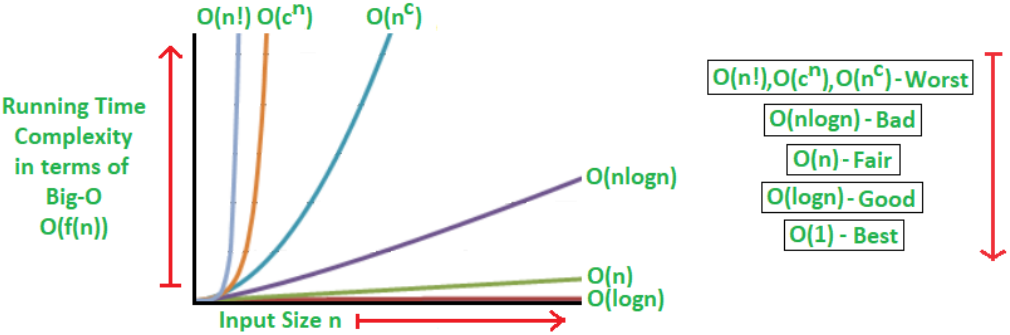
Problem Solutions

- Problem IV
  - Approach I: Linear search
    - Time complexity - $O(m \ x \ n)$
    - Space complexity - $O(n)$
  - Approach II: Binary search
    - Input list is already sorted
    - Time complexity - $O(m \ x \ log(n))$
    - One element can be searched in the list in logarithmic time - $O(log(n))$
    - Space complexity - $O(n)$
  - Approach III
    - Can we take advantage of the fact that both elements to be searched and list in which to search are sorted?
    - What would be the time complexity? O(n) or O(m)?
    - Does space complexity depend on order of list and elements to be searched in the input?

Problem Solutions

- Problem V
  - Selection sort
    - Time complexity - Quadratic - $O(n^2)$
    - Space complexity - Linear - O(n)
  - Yes, there are better sorting algorithms than selection sort
    - e.g. Heap sort, quick sort etc.
- Problem VI: matrix addition
  - Time complexity - Quadratic - $O(n^2)$
  - Space complexity - Quadratic - $O(n^2)$
- Problem VII: matrix multiplication
  - Time complexity - Cubic - $O(n^3)$
  - Space complexity - Quadratic - $O(n^2)$

Big Oh notations comparison



Source: https://www.geeksforgeeks.org/analysis-algorithms-big-o-analysis/