Asymptotic Analysis of Algorithms

Vaibhav Ranshoor [171080067]
Under guidance of: Pranav Nerurkar

Veermata Jijabai Technological institute

vaibhav.ranshoor1155@gmail.com

January 24, 2019

Table of Contents

Asymptotic Analysis

Asymptotic Analysis is the big idea that handles above issues in analyzing algorithms. In Asymptotic Analysis, we evaluate the performance of an algorithm in terms of input size (we dont measure the actual running time). We calculate, how does the time (or space) taken by an algorithm increases with the input size.

Different cases for Asymptotic analysis

We can have three cases for Asymptotic analysis of an algorithm:

- 1) Worst Case
- 2) Average Case
- 3) Best Case

Explanation of different cases for Asymptotic analysis

Worst Case

In the worst case analysis, we calculate upper bound on running time of an algorithm. We must know the case that causes maximum number of operations to be executed. For Eg.: The worst case time complexity of linear search would be O(n).

Average Case

In average case analysis, we take all possible inputs and calculate computing time for all of the inputs. Sum all the calculated values and divide the sum by total number of inputs. We must know (or predict) distribution of cases. For Eg.: In Linear search average case is O(n).

Best Case

In the best case analysis, we calculate lower bound on running time of an algorithm. We must know the case that causes minimum number of operations to be executed. For Eg.: The best case of linear search is O(1).

Different types of Asymptotic Notation

Types of Asymptotic Notation

1. Theta Notation
2. Big O Notation
3. Omega Notation

Asymptotic notations are mathematical tools to represent time complexity of algorithms for asymptotic analysis. The following 3 asymptotic notations are mostly used to represent time complexity of algorithms.

Asymptotic Notations

Name	**Symbol**	**Description**
Theta	Θ	Bound from above and below
Big O	O	Asymptotic Upper Bound
Omega	Ω	Asymptotic Lower Bound

Table: Table of Asymptotic Notations

Sorting Algorithms

Theorem (There are different types of sorting algorithms :)

Bubble sort

Insertion sort

Quick sort

Merge sort

Selection sort

Heap sort

Example (Bubble Sort)

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Example:

First Pass:

(5 1 4 2 8) to (1 5 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 is greater than 1.

(1 5 4 2 8) to (1 4 5 2 8), Swap since 5 is greater than 4

(1 4 5 2 8) to (1 4 2 5 8), Swap since 5 is greater than 2

(1 4 2 5 8) to (1 4 2 5 8), Now, since these elements are already in order (8 is greater than 5), algorithm does not swap them.

Sorting Algorithms

This illustration demonstrates working of bubble sort:

i = 0	j	0	1	2	3	4	5	6	7
	0	5	3	1	9	8	2	4	7
	1	3	5	1	9	8	2	4	7
	2	3	1	5	9	8	2	4	7
	3	3	1	5	9	8	2	4	7
	4	3	1	5	8	9	2	4	7
	5	3	1	5	8	2	9	4	7
	6	3	1	5	8	2	4	9	7
i = 1	0	3	1	5	8	2	4	7	9
	1	1	3	5	8	2	4	7	
	2	1	3	5	8	2	4	7	
	3	1	3	5	8	2	4	7	
	4	1	3	5	2	8	4	7	
	5	1	3	5	2	4	8	7	
i = 2	0	1	3	5	2	4	7	8	
	1	1	3	5	2	4	7		
	2	1	3	5	2	4	7		
	3	1	3	2	5	4	7		
	4	1	3	2	4	5	7		
i = 3	0	1	3	2	4	5	7		
	1	1	3	2	4	5			
	2	1	2	3	4	5			
	3	1	2	3	4	5			
i = 4	0	1	2	3	4	5			
	1	1	2	3	4				
	2	1	2	3	4				
i = 5	0	1	2	3	4				
	1	1	2	3					
i = 6	0	1	2	3					
		1	2						

Sorting Algorithms

Example (Insertion Sort)

Insertion sort is a simple sorting algorithm that works the way we sort playing cards in our hands.

Algorithm:

// Sort an arr[] of size n

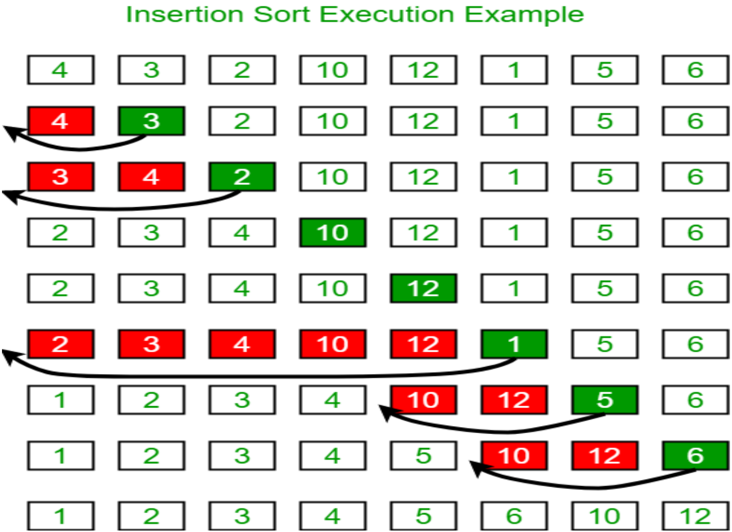
insertionSort(arr, n)

Loop from i = 1 to n-1.

a) Pick element arr[i] and insert it into sorted sequence arr[0i-1]

Sorting Algorithms

This illustration demonstrates working of Insertion sort:



Searching Algorithms

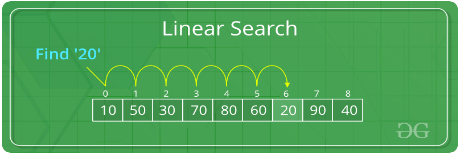
0.5 

Figure: This illustration demonstrates Linear Search

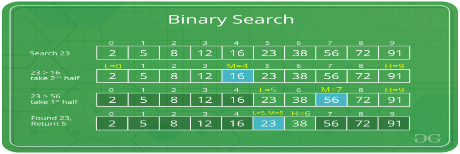
0.5 

Figure: This illustration demonstrates Binary Search

Figure: Different Searching Algorithms

References

www.geeksforgeeks.com (2019)
Computer Science Portal

The End