

Sunbeam Institute of Information Technology Pune and Karad

Module – Data Structures and Algorithms

Trainer - Devendra Dhande

Email - devendra.dhande@sunbeaminfo.com



Sunbeam Infotech

www.sunbeaminfo.co.

Algorithm Analysis

- · Analysis is done to determine how much resources it require.
- · Resources such as time or space
- There are two measures of doing analysis of any algorithm
 - · Space Complexity
 - Unit space to store the data into the memory (Input space) and additional space to process the data (Auxiliary space)
 - e.g. Algorithm to find sum of all array elements.

int arr[n] - n units of input space

sum, index, size - 3 units of auxiliary space

Total space required = input space + auxiliary space = n + 3 = n units

- · Time Complexity
 - · Unit time required to complete any algorithm
 - Approximate measure of time required to complete algorithm
 - · Depends on loops in the algorithm
 - Also depends on some external factors like type of machine, no of processed running on machine.
 - That's why we can not find exact time complexity.
- Method used to calculate complexities, is "Asymptotic Analysis"



Sunbeam Infotech

www.sunbeaminfo.com

Asymptotic Analysis

- It is a mathematical way to calculate complexities of an algorithm.
- It is a study of change in performance of the algorithm, with the change in the order of inputs.
- · It is not exact analysis
- Few mathematical notations are used to denote complexities.
- These notations are called as "Asymptotic notations" and are
 - Omega notation (Ω)
 - · Represents lower bound of the running algorithm
 - · It is used to indicate the best case complexity of an algorithm
 - Big Oh notation (O)
 - · Represents upper bound of the running algorithm
 - · It is used to indicate the worst case complexity of an algorithm
 - Theta notation (→)
 - Represents upper and lower bound of the running time of an algorithm (tight bound)
 - · It is used to indicate the average case complexity of an algorithm



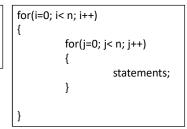
Sunbeam Infotech

www.sunbeaminfo.com

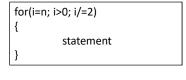
Time Complexity

Statement; constant

Linear



Quadratic



Logarithmic



Sunbeam Infotech

www.sunbeaminfo.com

Searching Algorithms : Time Complexity

Linear Search:

	No of Comparisons		Running Time	Time Complexity
Best Case	1	Key found at very first position	O(1)	O(1)
Average Case	n/2	Key found at in between position	O(n/2) = O(n)	O(n)
Worst Case	n	Key found at last position or not found	O(n)	O(n)

Binary Search:

	No of Comparisons		Running Time	Time Complexity
Best Case	1	Key found in very first iteration	O(1)	O(1)
Average Case	log n	Key found at non-leaf position	O(log n)	O(log n)
Worst Case	log n	if either key is not found or key is found at leaf position	O(log n)	O(log n)



Sunbeam Infotech

www.sunbeaminfo.com



Thank you!

Devendra Dhande devendra.dhande@sunbeaminfo.com/">devendra.dhande@sunbeaminfo.com/



Sunbeam Infotech

ww.sunbeaminfo.com