Bfs Dfs ComplexityAnalysis

Amith C A
2020MCS120003
DSC523 Project 1

Computational Complexity

Computational complexity is a field from computer science which analyzes algorithms based on the amount resources required for running it. The amount of required resources varies based on the input size, so the complexity is generally expressed as a function of n, where n is the size of the input.

It is important to note that when analyzing an algorithm we can consider the time complexity and space complexity. The space complexity is basically the amount of memory space required to solve a problem in relation to the input size. Even though the space complexity is important when analyzing an algorithm, in this story we will focus only on the time complexity.

Time Complexity

In computer science, the time complexity is the computational complexity that describes the amount of time it takes to run an algorithm. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, supposing that each elementary operation takes a fixed amount of time to perform.

Big-O Notation

Big-O notation, sometimes called asymptotic notation, is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.

In computer science, Big-O notation is used to classify algorithms according to how their running time or space requirements grow as the input size (n) grows. This notation characterizes functions according to their growth rates: different functions with the same growth rate may be represented using the same O notation.

Table of common time complexities

Name	**Time Complexity**
Constant Time	**O(1)**
Logarithmic Time	**O(log n)**
Linear Time	**O(n)**
Quasilinear Time	**O(n log n)**
Quadratic Time	**O(n^2)**
Exponential Time	**O(2^n)**
Factorial Time	**O(n!)**

Title: Bfs Dfs ComplexityAnalysis