Analysis of Algorithms | Set 4 (Analysis of Loops)

We have discussed Asymptotic Analysis, Worst, Average and Best Cases and Asymptotic Notations in previous posts. In this post, analysis of iterative programs with simple examples is discussed.

1) O(1): Time complexity of a function (or set of statements) is considered as O(1) if it doesn't contain loop, recursion and call to any other non-constant time function.

```
// set of non-recursive and non-loop statements
```

For example swap() function has O(1) time complexity.
A loop or recursion that runs a constant number of times is also considered as O(1). For example the following loop is O(1).

```
// Here c is a constant
for (int i = 1; i <= c; i++) {
    // some O(1) expressions
}
```

2) O(n): Time Complexity of a loop is considered as O(n) if the loop variables is incremented / decremented by a constant amount. For example following functions have O(n) time complexity.

```
// Here c is a positive integer constant
for (int i = 1; i <= n; i += c) {
    // some O(1) expressions
}

for (int i = n; i > 0; i -= c) {
    // some O(1) expressions
}
```

3) O(n^c): Time complexity of nested loops is equal to the number of times the innermost statement is executed. For example the following sample loops have O(n^2) time complexity

```
for (int i = 1; i <=n; i += c) {
   for (int j = 1; j <=n; j += c) {
     // some O(1) expressions
   }
}

for (int i = n; i > 0; i += c) {
   for (int j = i+1; j <=n; j += c) {
     // some O(1) expressions
}
```

For example Selection sort and Insertion Sort have O(n^2) time complexity.

4) O(Logn) Time Complexity of a loop is considered as O(Logn) if the loop variables is divided / multiplied by a constant amount.

```
for (int i = 1; i <=n; i *= c) {
   // some O(1) expressions
}
for (int i = n; i > 0; i /= c) {
   // some O(1) expressions
}
```

For example Binary Search(refer iterative implementation) has O(Logn) time complexity.

5) O(LogLogn) Time Complexity of a loop is considered as O(LogLogn) if the loop variables is reduced / increased exponentially by a constant amount.

```
// Here c is a constant greater than 1
for (int i = 2; i <=n; i = pow(i, c)) {
   // some O(1) expressions
}
//Here fun is sqrt or cuberoot or any other constant root
for (int i = n; i > 0; i = fun(i)) {
   // some O(1) expressions
}
```

See this for more explanation.

How to combine time complexities of consecutive loops?

When there are consecutive loops, we calculate time complexity as sum of time complexities of individual loops.

```
for (int i = 1; i <=m; i += c) {
    // some O(1) expressions
}
for (int i = 1; i <=n; i += c) {
    // some O(1) expressions
}
Time complexity of above code is O(m) + O(n) which is O(m+n)
If m == n, the time complexity becomes O(2n) which is O(n).
```

How to calculate time complexity when there are many if, else statements inside loops?

As discussed here, worst case time complexity is the most useful among best, average and worst. Therefore we need to consider worst case. We evaluate the situation when values in if-else conditions cause maximum number of statements to be executed.

For example consider the linear search function where we consider the case when element is present at the end or not present at all.

When the code is too complex to consider all if-else cases, we can get an upper bound by ignoring if else and other complex control statements.

How to calculate time complexity of recursive functions?

Time complexity of a recursive function can be written as a mathematical recurrence relation. To calculate time complexity, we must know how to solve recurrences. We will soon be discussing recurrence solving techniques as a separate post.

Quiz on Analysis of Algorithms

Next – Analysis of Algorithm | Set 4 (Solving Recurrences)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

Analysis Articles

Recommended Posts:

Analysis of Algorithm | Set 4 (Solving Recurrences)
Time Complexity of a Loop when Loop variable “Expands or Shrinks” exponentially
Analysis of Algorithms | Set 3 (Asymptotic Notations)
Analysis of Algorithms | Set 1 (Asymptotic Analysis)
Analysis of Algorithm | Set 5 (Amortized Analysis Introduction)

<< Previous Post Next Post >>

(Login to Rate and Mark)

1.3
Average Difficulty : **1.3/5.0**
Based on **97** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

Trending Content
Check for Majority Element in a sorted array
Enumeration of Binary Trees
Puzzle 67 | Fit Triangle
Java | Exception Handling | Question 3
GATE | GATE-CS-2003 | Question 18
Rearrange array in alternating positive & negative items with O(1) extra space | Set 1
Setting up the environment in Java
Swap Kth node from beginning with Kth node from end in a Linked List
Point arbit pointer to greatest value right side node in a linked list
k largest(or smallest) elements in an array | added Min Heap method

ProGeek Cup 1.0

Most Visited Posts
Top 10 Algorithms and Data Structures for Competitive Programming
Top 10 algorithms in Interview Questions
How to begin with Competitive Programming?
Step by Step Guide for Placement Preparation
How to prepare for ACM-ICPC?
Insertion Sort , Binary Search , QuickSort , MergeSort , HeapSort

ads by BSA

Popular Categories
Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Geometric Algorithms
Searching
Sorting
Analysis of Algorithms

Popular Categories
Mathematical Algorithms
Randomized Algorithms
Recursion
Game Theory

Tags

Advanced Data Structure Amazon Aptitude Aptitude Arrays Bit Magic C C C++ C++ Quiz CPP-Library C Quiz Data Structures Data Structures DBMS Dynamic Programming Experienced GATE-CS-2012 GBlog Graph Hash Internship Interview Experiences Java java- Java Quiz Linked Lists Mathematical Matrix MCQ Microsoft number-digits Program Output Project Puzzles Python QA - Placement Quizzes QA - Placement Quizzes School Programming Searching Sorting STL Strings Technical Scripter Trees

Advertise Here

Recent Comments

@geeksforgeeks, Some rights reserved Contact Us! About Us! Careers! Privacy Policy