GeeksforGeeks
A computer science portal for geeks

GeeksQuiz

Login

Home Algorithms DS GATE Interview Corner Q&A C C++ Java Books Contribute Ask a Q About

Array Bit Magic C/C++ Articles GFacts Linked List MCQ Misc Output String Tree Graph

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
   }
```

Google™ Custom Search

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
   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

Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding "extern" keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

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).
```

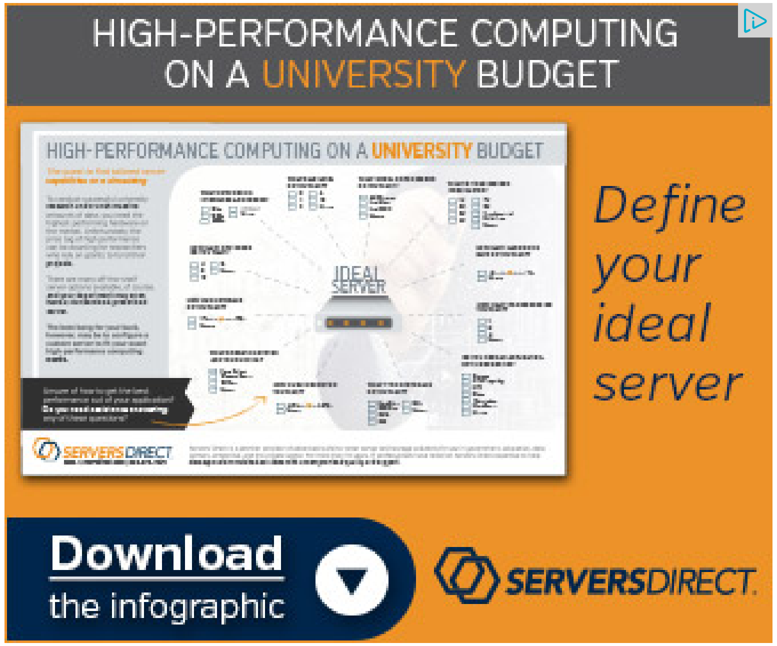
How to calculate time complexity when there are many if, else statements inside loops?
As discussed here, worst case time complexity is the most useful among best, average and worst. Therefore we need to consider worst case. We evaluate the situation when values in if-else conditions cause maximum number 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

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



Related Tpoics:

- Analysis of Algorithms | Set 3 (Asymptotic Notations)
- NP-Completeness | Set 1 (Introduction)
- Static and Dynamic Libraries | Set 1
- The Ubiquitous Binary Search | Set 1
- Reservoir Sampling

705

Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 29 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 32 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 57 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 59 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

- Analysis of Algorithms | Set 2 (Worst, Average and Best Cases)
- Analysis of Algorithms | Set 1 (Asymptotic Analysis)
- Scope rules in C

12 Tweet 4 1

Writing code in comment? Please use **ideone.com** and share the link here.

17 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion…

Sahil • 6 days ago

can we say that the complexity of log2() of c++ in math library is constant or s

Vishal • 13 days ago

Amazing post

Ganesh • 18 days ago

In 1st and 2nd,
If value of C is 1 the complexity is O(1) and if value of C is 2 then complexity is
consider value of c = 1 and say complexity is O(n)

mak ➦ Ganesh • 11 days ago

No, C is a constant. So, every time the code runs, the value of c will re
cases will have C=1. Its not possible that once run has C=1 and other

Ganesh ➦ mak • 10 days ago

Thanks Mak for your reply but I guess I just got confused.
Let me repharse my question

what will be the complexity of following

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

Here if we consider the (1) and (2) will it be O(1) or O(2)?
The only diff I see in (1) and (2) is they say if n is constant then contradicts that it says complexity will be O(n).
Could you please clarify?

sagar • a month ago

in 3rd one if c=3..then complexity is O(n^3)...?

mak ➦ sagar • 11 days ago

No, it will still be the same.

Guest • a month ago

for(i=n/2;i<n;i++) {="" for(j="i+1;j&lt;n;j++)" {="" k="k+n/2;" }="" }="" is="" this="

deepika • a month ago

for(i=n/2;i<n;i++) {="" for(j="i+1;j&lt;n;j++)" {="" k="k+n/2;" }="" }="" is="" this="
help="" me,="" am="" i="" right="" ??="">

nikeadam • 3 months ago

```
//as u mentioned this as O(n)
for (int i = 1; i <= n; i += c) { //c is any positive integer
// some O(1) expressions
}

//and this as O(1)
for (int i = 1; i <= c; i++) {
// some O(1) expressions
}
```

what if c=1?? both are same, how does both differ in time complexity

GeeksforGeeks Mod nikeadam • 3 months ago

nikeadam, please take a closer look the first loop runs O(n) times, but is same as O(1) for a constant c.

hari GeeksforGeeks • 3 months ago

So, It depends on the loop variable. If the loop variable is incren
factor and if the loop runs for 'n' times, then it is O(n).
At the same time, If the loop runs for 'n' times with constant inc
Is this right ?

Utkarsh Gupta • 3 months ago

Asymptotic notations are for performance analysis that is abstract measurem swaps / operation). Asymptotic notations are usually used for extremely larger key from a thousand OR a million.

So if n (input) is very large and variable then the complexity of time (measured operation) will be in terms of n. But if the instructions are there in a loop with fi

operation) will be in terms of n. But if the instructions are there in a loop with n,
will be O(1).

Gaurav pruthi • 3 months ago

"A loop or recursion that runs a constant number of times is also considered a
Is this statement true ? ...if yes then how
beacuse complexity will be O(c) and whats the difference between O(n) and O

Sudheer Reddy Jakkam ➦ Gaurav pruthi • 3 months ago

Big O notation ignores constants.
O(log n) is exactly the same as O(log(n^c)). The logarithms differ only
notation ignores that. Similarly, logs with different constant bases are e

refer: http://web.mit.edu/16.070/www/...

Kartik ➦ Gaurav pruthi • 3 months ago

Gaurav, please note that O(1) means a constant. So O(2), O(3) or O(c
me if I am wrong.

1

Jayash • 3 months ago

very helpful

Subscribe   Add Disqus to your site