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

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

Related Topics:

- An interesting time complexity question
- A Problem in Many Binary Search Implementations
- 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
- Analysis of Algorithms | Set 2 (Worst, Average and Best Cases)

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

msk • a month ago

Can some one tell what would be the complexity of

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

• Reply • Share ›

lhr88 msk • a month ago

I think the complexity would depend on the variable p (that is how p changes within the loop).
It should be O(n) if p is added/subtracted.
It should be O(log n) if p is multiplied/divided my a constant amount

• Reply • Share ›

Paramanand • 4 months ago

can you explain the difference between O(n) and O(1) still clearly

• Reply • Share ›

anshika sharma Paramanand • 4 months ago

O(n) means the time complexity of algorithm is a linear function of n(input size)
O(1) means the time complexity of the algorithm is always constant!

1 • Reply • Share ›

MrGrj • 4 months ago

Can you give me an example of O(n^m) ?

• Reply • Share ›

aNewBornCoder MrGrj • 2 months ago

m nested loops and every loop is for(i=0;i<n;i++){} .="">

1 • Reply • Share ›

Mohamed Naser • 8 months ago

it still O(n) if c >= 1 or c > 1 only

• Reply • Share ›

Guest • 8 months ago

Which search method will have less time taking for an unsorted array?
I think it should be Linear.

• Reply • Share ›

John Carter • 9 months ago

3) O(nc): In this section the second example
should be as below

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

1 • Reply • Share ›

Rakshita Kaulgud ➚ John Carter • 8 months ago

in the above code, the inner loop wont get executed because the value of j is n+1 (because i = n) and the condition is false at the beginning itself...

• Reply • Share ›

iowa • 10 months ago

what would be time complexity if you have two outer for loop running n times and one for loop inside second for loop running 3 times ?
will it be O(n^2)+O(3) or O(n^2) ????

• Reply • Share ›

Praveen Kumar ➚ iowa • 8 months ago

The expression O(3) is not valid and doesn't have a meaning. Although one would easily infer that you mean O(1). When we do asymptotic analysis we drop the smaller terms because they become insignificant when n becomes very large. Here , the second for loop running 3 times would add constant overhead , which can be represented as O(1) so exact time complexity would be O(n^2)+ O(1) but we should ignore O(1) in comparison to O(n^2) so the time complexity would be O(n^2) , it's sufficient to describe the time complexity of your program.

• Reply • Share ›

Mohammad Abul Khaer ➚ iowa • 10 months ago

O(n^2)

• Reply • Share ›

gayu • a year ago

what is the time compexity of the following..?
for(int i=1;i<n;i++) o(n)="" or="" o(1)...?="" bcoz="" herei++="" means="" i+="1....">

• Reply • Share ›

Bharti gayu • 10 months ago

o(n)

• Reply • Share ›

lk15198 • a year ago

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
}
```

Here the correction would be if the number of times loop iterate depend upon size of input then it would be O(n).

Please correct if am wrong?

3 • Reply • Share ›

Praveen Kumar lk15198 • 8 months ago

Yes , you are wrong. Asymptotic analysis in itself is how the running time depends on the size of input , here the definition is correct , as long as the loop variable is incremented/decremented by a constant amount the time complexity would be O(n).

• Reply • Share ›

kk lk15198 • 9 months ago

well, if instead of i++ in the for loop , if we do i=i*2, then also it depend on the size of input but its not o(n)

• Reply • Share ›

Sahil • a year ago

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

Vishal • a year ago

Amazing post

Guest • a year 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 O(n). Is it correct? Or can we consider value of c = 1 and say complexity is O(n)

mak ➚ Guest • a year ago

No, C is a constant. So, every time the code runs, the value of c will remain same. So, if C=1, then all the cases will have C=1. Its not possible that once run has C=1 and other has C=2..

Ganesh ➚ mak • a year 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 complexity will be O(1) but (2) contradicts that it says complexity will be O(n).
Could you please clarify?

ashish singla ➚ Ganesh • 7 months ago

in (1) c can be any costant but has to be a constant.

your input will not change the number of times loop needs to run.

But in (2) n is changing(but c must be constant), which will decide how many times your loop needs to run. In (2) it runs of loop will change as your input changes.

1

sagar • a year ago

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

• Reply • Share ›

mak ➚ sagar • a year ago

No, it will still be the same.

• Reply • Share ›

bumba ➚ mak • a year ago

it will be O(n^3) when code will be like
for(i=1;i<n;i++){ for(j="1;j<n;j++){" for(k="1;k<n;k++){statement};" }="" }="">

• Reply • Share ›

Guest • a year ago

for(i=n/2;i<n;i++) {="" for(j="i+1;j<n;j++)" {="" k="k+n/2;" }="" }="" is="" this="" o(n^2)="" ?="">

• Reply • Share ›

deepika • a year ago

for(i=n/2;i<n;i++) {="" for(j="i+1;j<n;j++)" {="" k="k+n/2;" }="" }="" is="" this="" o(n^2)="" ??="" can="" any="" one="" help="" me,="" am="" i="" right="" ??="">

• Reply • Share ›

hanuman_patel ➚ deepika • 7 months ago

0((n^2)/2)=0(n^2)

• Reply • Share ›

nikeadam • a year 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

• Reply • Share ›

GeeksforGeeks Mod ➚ nikeadam • a year ago

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

• Reply • Share ›

hari ➚ GeeksforGeeks • a year ago

So, It depends on the loop variable. If the loop variable is incremented/decremented by a constant 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 increment/ decrements of 1, it is O(1).
Is this right ?

1 • Reply • Share ›

raveena ➚ hari • 8 months ago

in both cases time complexity is o(n)
time complexity is o(1) when loop runs for some constant times...i.e.for(i=1;i<=c;i++) in this case time complexity is o(1)
plz correct if i m wrong.....

• Reply • Share ›

Bharti ➚ hari • a year ago

i think in both cases the complexity is o(n)

• Reply • Share ›

Utkarsh Gupta • a year ago

Asymptotic notations are for performance analysis that is abstract measurement of time (rather comparisons / swaps / operation). Asymptotic notations are usually used for extremely larger values of input size e.g. searching a key from a thousand OR a million.

So if n (input) is very large and variable then the complexity of time (measured as number of comparisons / swaps / operation) will be in terms of n. But if the instructions are there in a loop with fixed constant size then the complexity will be O(1).

• Reply • Share ›

Gaurav pruthi • a year ago

"A loop or recursion that runs a constant number of times is also considered as O(1)"
Is this statement true ? ...if yes then how
beacuse complexity will be O(c) and whats the difference between O(n) and O(c) since both are linear eqns..

• Reply • Share ›

Sudheer Reddy Jakkam ➚ Gaurav pruthi • a year ago

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

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

Kartik ➚ Gaurav pruthi • a year ago

Gaurav, please note that O(1) means a constant. So O(2), O(3) or O(c) all are same as O(1). Please correct me if I am wrong.

chetan ➚ Kartik • a year ago

C means constant, if C = 5 it remains same through out the application and it never changes.
Ex 1:Running a loop

```
for(int i = 0; i <= C; i++)
{
//Executes C times, any time you run your application it //remains same
}
```

so the complexity is O(C), Usually in Algo analysis constant are set as 1 so the complexity is O(1)

For Ex 2: Linear Searching

```
int a[N] = {/*Some Values*/};
for(int i = 0; i < N; i++)
{
//Sorting Logic: Here searching depends upon no. of //elements in the array and it may changes each time //when you run your application
}
```

so the complexity varies depending upon the no. of elements in the array which increases the loop so the complexity is O(N)

Jayash • a year ago

very helpful

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

- Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 1 hour ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 2 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 2 hours ago