GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- 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 2 (Worst, Average and Best Cases)

In the previous post, we discussed how Asymptotic analysis overcomes the problems of naive way of analyzing algorithms. In this post, we will take an example of Linear Search and analyze it using Asymptotic analysis.

We can have three cases to analyze an algorithm:
1) Worst Case
2) Average Case
3) Best Case

Let us consider the following implementation of Linear Search.

```
#include <stdio.h>
```

```
// Linearly search x in arr[].  If x is present then return the index,
// otherwise return -1
int search(int arr[], int n, int x)
{
    int i;
    for (i=0; i<n; i++)
    {
       if (arr[i] == x)
         return i;
    }
    return -1;
}

/* Driver program to test above functions*/
int main()
{
    int arr[] = {1, 10, 30, 15};
    int x = 30;
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("%d is present at index %d", x, search(arr, n, x));

    getchar();
    return 0;
}
```

Worst Case Analysis (Usually Done)
In the worst case analysis, we calculate upper bound on running time of an algorithm. We must know the case that causes maximum number of operations to be executed. For Linear Search, the worst case happens when the element to be searched (x in the above code) is not present in the array. When x is not present, the search() functions compares it with all the elements of arr[] one by one. Therefore, the worst case time complexity of linear search would be [Tex]\theta(n)[/tex].

Average Case Analysis (Sometimes done)
In average case analysis, we take all possible inputs and calculate computing time for all of the inputs. Sum all the calculated values and divide the sum by total number of inputs. We must know (or predict) distribution of cases. For the linear search problem, let us assume that all cases are uniformly distributed (including the case of x not being present in array). So we sum all the cases and divide the sum by (n+1). Following is the value of average case time complexity.

```
Average Case Time = [Tex]{\sum_{i=1}^{n+1} \theta (i)}\over{(n+1)}[/Tex]

                  = [Tex]{ \theta ((n+1)*(n+2)/2)}\over{(n+1)}[/Tex]

                  = [Tex]{ \theta (n)[/Tex]
```

Best Case Analysis (Bogus)
In the best case analysis, we calculate lower bound on running time of an algorithm. We must know the case that causes minimum number of operations to be executed. In the linear search problem, the best case occurs when x is present at the first location. The number of operations in the best case is constant (not dependent on n). So time complexity in the best case would be [Tex]\theta(1)[/tex]

Most of the times, we do worst case analysis to analyze algorithms. In the worst analysis, we guarantee an upper bound on the running time of an algorithm which is good information.
The average case analysis is not easy to do in most of the practical cases and it is rarely done. In the average case analysis, we must know (or predict) the mathematical distribution of all possible inputs.
The Best Case analysis is bogus. Guaranteeing a lower bound on an algorithm doesn't provide any information as in the worst case, an algorithm may take years to run.

For some algorithms, all the cases are asymptotically same, i.e., there are no worst and best cases. For example, Merge Sort. Merge Sort does [Tex]\theta(nLogn)[/tex]operations in all cases. Most of the other sorting algorithms have worst and best cases. For example, in the typical implementation of Quick Sort (where pivot is chosen as a corner element), the worst occurs when the input array is already sorted and the best occur when the pivot elements always divide array in two halves. For insertion sort, the worst case occurs when the array is reverse sorted and the best case occurs when the array is sorted in the same order as output.

References:
MIT's Video lecture 1 on Introduction to 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 4 (Analysis of Loops)
- 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

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

Sriram · 5 months ago
If a large problem is given with two solution of time complexity theta(n) and w(n) which one should be selected.

• Reply • Share ›

tvs551 Sriram • 3 months ago

we should consider theta(n) because w(n) is consider as lower bound running time.

• Reply • Share ›

evolving • 8 months ago

this post uses Theta notation for asymptotic analysis while in general all algorithms involve time complexity in big O notation . Plz explain ur reason for using Theta notaion

13 • Reply • Share ›

Adauta Garcia Ariel evolving • 4 months ago

If we analize the algorithm for linear search, counting the operations:

f(n) = 3+5n In the worst case.

So we can say that we have found the tight bound, we write it :

f(n) is theta(n).

You use Big O for upper bound, so you can write it: O(n)

Even O(n^2) , but it is less precise.

Big O is used to describe the behavior of our algorithm and know that will never exceed a certain bound.

The most of time is a little more hard to find the theta notation, so we use O notation.

Here for more info about it.

• Reply • Share ›

Guest Adauta Garcia Ariel • 3 months ago

thanks for this http://discrete.gr/complexity/ :)

• Reply • Share ›

VAIBHAV GUPTA • a year ago

In last line of "Best Case Analysis (Bogus)" (The number of operations in worst case is constant (not dependent on n).) It should be best case and not worst case.

11 • Reply • Share ›

GeeksforGeeks Mod VAIBHAV GUPTA • 7 months ago

Thanks for pointing this out. We have corrected the typo.

• Reply • Share ›

harshal kokate • a year ago

upper bound and lower bound ? n diffrence in upper and worst case ?

^ | v • Reply • Share ›

Marsha Donna • a year ago

why does (n-1)! have lower order of growth compared to n!

5 ^ | v • Reply • Share ›

Jonathan → Marsha Donna • a year ago

They have the same order of growth.

(n-1)! / n! as n->infinity ... the minus 1 becomes irrelevant.

Thus, we get n!/n! = 1.

3 ^ | v • Reply • Share ›

Ravi Teja Kaveti → Jonathan • 10 months ago

limit (n->infinity) (n-1)!/n! = 0 which signifies (n-1)! as lower growth rate compared to n!

^ | v • Reply • Share ›

Marsha Donna • a year ago

the average case analysis expression 4 linear search evaluates to {theta(n^2)}/(n+1) which is not same as theta(n) please clarify???

1 ^ | v • Reply • Share ›

varahi → Marsha Donna • a year ago

its same

1 ^ | v • Reply • Share ›

Marsha Donna • a year ago

the average case analysis expression is given as = so it evaluates to {\theta(n^2)/(n+1) } which is not same as theta(n) please clarify???

^ | v • Reply • Share ›

Jonathan → Marsha Donna • a year ago

When performing algorithmic analysis, you are looking at cases where n is a VERY VERY BIG number (thus, think of n->infinity).

So, for (n^2)/(n+1) ... what happens when n is a gigantic number? Well, the '+ 1' is essentially redundant right? If n was some small number like 2, then n+1 = 3 may be significant.

However, if n was 5000000. Well, adding 1 and getting 5000001 doesn't make a big difference at all right? So, we just ignore it.

We end up with theta((n^2)/(n)) which simplifies to theta(n)

2 • Reply • Share ›

Sanjay • a year ago

Shouldn't the best case for quick sort be the case when the array is already sorted in the order the quick-sort has to sort ? As the sort will have nothing to do (e.g. swapping elements) as all elements lying to the right of the pivot element will always remain to the right as they are greater than the pivot element (assuming ascending order and pivot element to be the first element of the array).

• Reply • Share ›

ak Sanjay • a year ago

No the quicksort does not have already sorted array as a best case , It can be a best case scenario only if we divide the array into two equal halves.

Try dividing the array(by choosing pivot) as
T(n)=T(n-1)+T(1)
after every iteration.

So For a quicksort Time required is dependent on how you choose the pivot and not on the order of elements.

1 • Reply • Share ›

Robin Thomas • 2 years ago

I would like to point out the fact that the worst case condition of linear search is when the element to be searched is not in the array, or when its the last element of the array. In either case, the running time shall be O(n).

(You had mentioned the first case, but not the second one)

1 • Reply • Share ›

anon • 2 years ago

shouldn't the worst case for linear search be O(n) and not ?(n)?

• Reply • Share ›

Robin Thomas anon • 2 years ago

Actually both are true. O(n) and ?(n) shall be the same, since the latter is more strictly bound. Big O is not the only class in asymptotic. There are four more classes too, and big ? is another one of it.

• Reply • Share ›

Ayesha Karim • 2 years ago

http://www.cricketoverflow.com

• Reply • Share ›

Poorna Durga Yeddu • 2 years ago

Why merge sort takes O(nlogn)operations in all cases?

1 • Reply • Share ›

GeeksforGeeks → Poorna Durga Yeddu • 2 years ago

Take a closer look at the merge process of merge sort, it always takes *theta(n)* time. Therefore, recurrence of Merge sort is always following.

T(n) = T(n/2) + *theta(n)*

The solution of above recurrence is *theta(nLogn)*

2 • Reply • Share ›

Poorna Durga Yeddu → GeeksforGeeks • 2 years ago

thanks

1 • Reply • Share ›

Poorna Durga Yeddu → GeeksforGeeks • 2 years ago

I understood thank you

2 • Reply • Share ›

Poorna Durga Yeddu • 2 years ago

So, can I conclude that worst case analysis is best for algorithm analysis?

2 • Reply • Share ›

GeeksforGeeks → Poorna Durga Yeddu • 2 years ago

yes, most of the times.

1 • Reply • Share ›

Suraj Prakash Sahu • 2 years ago

thanks friends

• Reply • Share ›

vkjk89 • 3 years ago

Nice one. :)
Could you please have one tutorial on the 3 notations used for complexity analysis i.e. (Theta,Ohm and Big-O) ?
Whats exact difference between these and which to use when ?

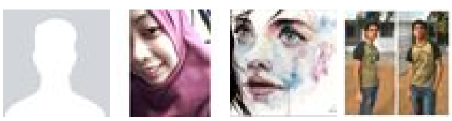
21 • Reply • Share ›

GeeksforGeeks Mod → vkjk89 • 7 months ago

A Tutorial for asymptotic notations has been published here (

http://www.geeksforgeeks.org/a...)

Reply • Share ›



- 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