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.

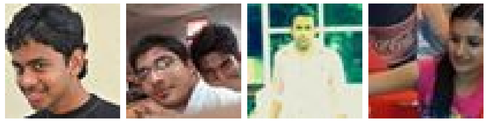
```
#include <stdio.h>

// 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()
{
```

Google™ Custom Search

GeeksforGeeks

53,522 people like GeeksforGeeks.

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



```
    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 $\theta(n)$.

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.

$$\text{Average Case Time} = \frac{\sum_{i=1}^{n+1} \theta(i)}{(n+1)}$$

$$= \frac{\theta((n+1)*(n+2)/2)}{(n+1)}$$

$$= \theta(n)$$

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 worst case is constant (not dependent on n). So time complexity in the best case would be $\theta(1)$

Geometric Algorithms

Popular Posts

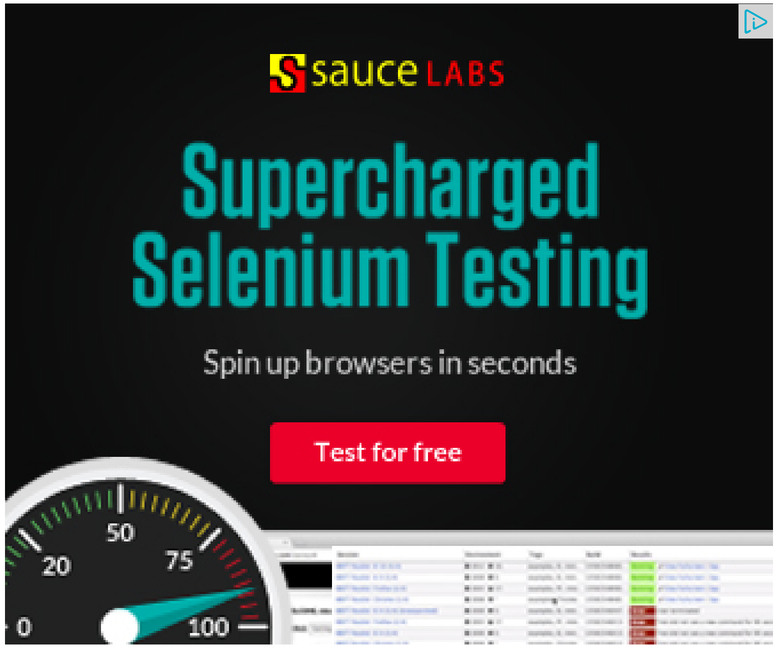
- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common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

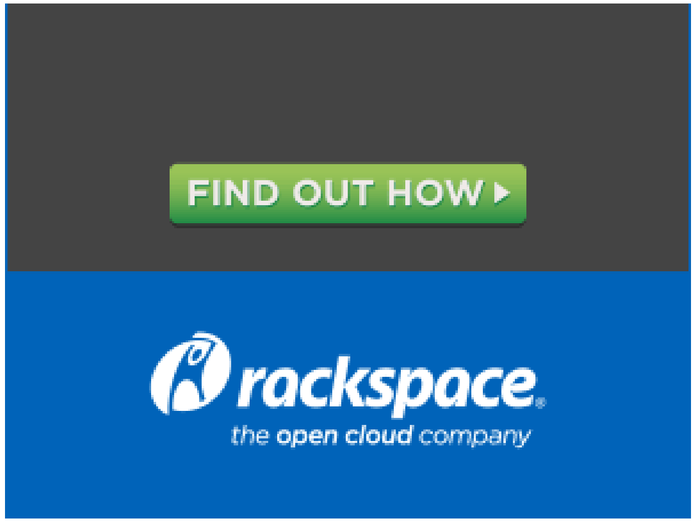
Most of the times, we do worst case analysis to analyze algorithms. In the worst analysis, we guarantee an upper bound on the running time of an algorithm which is good information.
The average case analysis is not easy to do in most of the practical cases and it is rarely done. In the average case analysis, 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 $\theta(nLogn)$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.





705 Subscribe

Related Tpoics:

- 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
- Analysis of Algorithms | Set 1 (Asymptotic Analysis)
- Scope rules in C

11 Tweet 0 3

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

20 Comments GeeksforGeeks

Sort by Newest

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

harshal kokate · 8 days ago

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

Marsha Donna · 4 months ago

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

2

Jonathan Marsha Donna · 2 months 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.

Marsha Donna · 4 months 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

varahi Marsha Donna · a month ago

its same

Marsha Donna · 4 months ago

the average case analysis expression is given as = so it evaluates to {\theta(n
theta(n) please clarify???

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

Jonathan ➜ Marsha Donna • 2 months ago

When performing algorithmic analysis, you are looking at cases where
think of n->infinity).

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

However, if n was 5000000. Well, adding 1 and getting 5000001 doesn
we just ignore it.

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

Sanjay • 4 months 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).

Robin Thomas • 9 months ago

I would like to point out the fact that the worst case condition of linear search is
not in the array, or when its the last element of the array. In either case, the ru

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

anon • 10 months ago

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

Robin Thomas ➜ anon • 9 months ago

Actually both are true. O(n) and ?(n) shall be the same, since the latter
only class in asymptotic. There are four more classes too, and big ? is

Ayesha Karim • a year ago

http://www.cricketoverflow.com

Poorna Durga Yeddu • a year ago

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

GeeksforGeeks ➜ Poorna Durga Yeddu • a year ago

Take a closer look at the merge process of merge sort, it always takes
of Merge sort is always following.

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

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

Poorna Durga Yeddu ➜ GeeksforGeeks • a year ago

thanks

Poorna Durga Yeddu ➜ GeeksforGeeks • a year ago

I understood thank you

Poorna Durga Yeddu • a year ago

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

GeeksforGeeks → Poorna Durga Yeddu • a year ago

yes, most of the times.

Suraj Prakash Sahu • a year ago

thanks friends

vkjk89 • 2 years ago

Nice one. :)

Could you please have one tutorial on the 3 notations used for complexity anal

Whats exact difference between these and which to use when ?

18

Subscribe Add Disqus to your site