Analysis of Algorithms | Set 1 (Asymptotic Analysis)

Why performance analysis?

There are many important things that should be taken care of, like user friendliness, modularity, security, maintainability, etc. Why to worry about performance?

The answer to this is simple, we can have all the above things only if we have performance. So performance is like currency through which we can buy all the above things. Another reason for studying performance is – speed is fun!

Given two algorithms for a task, how do we find out which one is better?

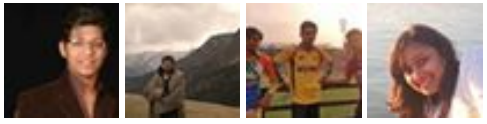
One naive way of doing this is – implement both the algorithms and run the two programs on your computer for different inputs and see which one takes less time. There are many problems with this approach for analysis of algorithms.

1) It might be possible that for some inputs, first algorithm performs better than the second. And for some inputs second performs better.

2) It might also be possible that for some inputs, first algorithm perform better on one machine and the second works better on other machine for some other inputs.

Asymptotic Analysis is the big idea that handles above issues in analyzing algorithms. In Asymptotic Analysis, we evaluate the performance of an algorithm in terms of input size (we don't measure the actual running time). We calculate, how does the time (or space) taken by an algorithm increases with the input size.

For example, let us consider the search problem (searching a given item) in a sorted array. One way to search is Linear Search (order of growth is linear) and other way is Binary Search (order of growth is logarithmic). To understand how Asymptotic Analysis solves the above mentioned problems in analyzing algorithms, let us say we run the Linear Search on a fast computer and Binary Search on a slow computer. For small values of input array size n, the fast computer may



take less time. But, after certain value of input array size, the Binary Search will definitely start taking less time compared to the Linear Search even though the Binary Search is being run on a slow machine. The reason is the order of growth of Binary Search with respect to input size logarithmic while the order of growth of Linear Search is linear. So the machine dependent constants can always be ignored after certain values of input size.

Does Asymptotic Analysis always work?
Asymptotic Analysis is not perfect, but that's the best way available for analyzing algorithms. For example, say there are two sorting algorithms that take 1000nLogn and 2nLogn time respectively on a machine. Both of these algorithms are asymptotically same (order of growth is nLogn). So, With Asymptotic Analysis, we can't judge which one is better as we ignore constants in Asymptotic Analysis.
Also, in Asymptotic analysis, we always talk about input sizes larger than a constant value. It might be possible that those large inputs are never given to your software and an algorithm which is asymptotically slower, always performs better for your particular situation. So, you may end up choosing an algorithm that is Asymptotically slower but faster for your software.

We will covering more on analysis of algorithms in some more posts on this topic.

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.

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding "extern" keyword in C
Median of two sorted arrays

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 2 (Worst, Average and Best Cases)
- Scope rules in C

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

15 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

aRUN • 21 days ago

PLZ TELL ME THE TIME COMPLEXITY OF FOLLOWING PROGRAM IT'S KI

```
#include <c:\opencv\build\include\opencv\cvaux.h>

#include <c:\opencv\build\include\opencv\highgui.h>

#include <c:\opencv\build\include\opencv\cxcore.h>
```

```
#include <omp.h>

#include <time.h>

#include <iostream>

#include <fstream>

#include <cmath>

#include <stdio.h>

#include <stdlib.h>
```

see more

1 ▲ | ▼ •

Jova • a month ago

Can you clarify the statement:
"we evaluate the performance of an algorithm in terms of input size (we don't r
you clarify what running time is? Running time in other books mean steps or o

▲ | ▼ •

harsha ➚ Jova • 13 days ago

Running time is time taken by program to run and display desired outp
up which algorithm is better. Asymptotic notation doesn't give actual ru
is increased with respect to size of input .

▲ | ▼ •

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

Lily • 3 months ago

very helpful website..

hari • 5 months ago

useful

Rajavenu Kyatham • 10 months ago

awesome website....

Kush Pandey • 11 months ago

good job

hxgxs1 • a year ago

Doubt!!!

It says that

"in Asymptotic analysis, we always talk about input sizes larger than a constar

In the case of 1000nlogn and 2nlogn the constant values are 1000 and 2 right?

10

Robin Thomas hxgxs1 • 9 months ago

Nope. 1000 and 2 are merely machine dependent constants. Those ar

In asymptotic analysis, we always assume the input size n >= n0, whe

formal definition of big O, or big Omega or the other classes, you shall

4

hari Robin Thomas • 5 months ago

you are right man

Sampat S Magi • a year ago

great site. knowledgeable.

Deepak Singh • a year ago

great work

Ash • a year ago

Didn't you mean "For small values of input array size n, the *fast* computer m

GeeksforGeeks ➦ Ash • a year ago

Thanks for pointing this out. There was a typo. We have corrected the

1 

latha • 2 years ago

nice one thank u:)