In21-S2-CS2023 - Data Structures and Algorithms

Complexity Analysis

Why should we care about complexity analysis?

We want to analyze few things.

- How an alagorithm is affected when we change the size of the input.

Ex: For insertion sort, we when double the size of the input array, how does the performance gets affected?

- How to compare multiple possible algorithms for the same task?

Ex: To find whether a given number N is a prime, we can think of two ways

- 1st method: Divide N from the numbers 1 to N. If only 2 numbers divides N, return N as a prime.

- 2nd method: Divide N from the number 1 to sqrt(N). If only 1 divides N, return N as a prime.

We know 2nd method is "better". But why?

Ok, so we can just measure the time spent...

But different computers vary in

- CPU speed
- memory
- effect on background processes
- ...

We can not do a "perfect" experiment to analyze the quality or the performance of an algorithm. We need a method which does not depend on the computer or the execution environment to analyze the algorithms.

So we use compelixty analysis (i.e. asymptotic analysis) to evaulate and compare the algorithms.

Main idea: Ignore the actual time spent or memory needed. Focus on the growth of time and memory as a function of the input size (n).

What is Time Complexity Analysis? - Basics of Algorithms ⏳

Previous activity

◄ Lecture slides II

Jump to...

Next activity

Complexity Analysis - Take home assignment ►

Stay in touch

University of Moratuwa

https://uom.lk

0094 11 26 400 51

info[AT]uom[.]lk

Counting Steps

Before directly going in to asymptotic analysis, we can analyze an algorithm (for its time complexity) by counting the total steps needed to complete the algorithm for a given input size.

Please refer the following video to learn about this.

https://mediaspace.msu.edu/media/CSE+431A+Exact+Algorithm+Analysis/1_j42i0mm1

```
1       function ArrayMax(A, n)
2           current_max = -infinity
3           i = 1
4           while i<=n do
5               if current_max < A[i] then
6                   current_max = A[i]
7               i++
8           return current_max
```

Refer to the above ArrayMax algorithm. Calculate the minimum and the maximum number of **times** each line will execute. Assume that all the elements in the array ara finite numbers.

line	minimum	maximum
1		1
2		n
3		n+1
4		1
5		1
6		n
7		1
8		1
		n
		1
		1
		n
		n
		n+1

Check

Why Complexity Analysis

2 / 5

Reuse Rights of use Embed