Universitat de Lleida

Measure Algorithms Efficiency

Jordi Planes & Aitor Corchero

Universitat de Lleida

15-02-2016

Index

Algorithms

Algorithms Definition

An algorithm is a procedure that takes a value or set of values as input and produces a value or a set of values as an output.
An algorithm is a set of instructions that solves certain problem.

Algorithms

Is sufficient the algorithm correctness?? NO

Algorithms

Is sufficient the algorithm correctness?? NO

- The Algorithms has to respond on time (Time metrics)
- The algorithms has to use lesser resources as possible (Space metrics)

Algorithms Performance (Time vs Space)

TIME	**SPACE**
Instructions take time	Data structures take space
Algorithm perform speed?	Data structures to be used?
What affects its runtime?	How data structure affect the runtime?

Algorithms Performance (Time vs Space)

TIME	SPACE
Instructions take time	Data structures take space
Algorithm perform speed?	Data structures to be used?
What affects its runtime?	How data structure affect the runtime?

Time vs Space

The best solution is:

- Select **efficient data structures**
- Use **efficient methods** over the selected data structures.

How we can measure the Time/Space efficiency?

Measuring Cost

Cost Definition

The Cost (C) permit to benchmark different algorithms by considering a specific metric.

metric = Time

How to measure algorithm time??

Measuring Cost

Example 1: If structure

Calculate the cost in an if structure

Measuring Cost

```
if n%2==0:
    print "Odd!!"
else:
        print "Even"
```

Cost (C_i)	**Time (T_i)**
C_1	1
C_2	1
C_3	1

$$Cost = C_1 + max(C_2, C_3) = 2$$

Measuring Cost

Example 2: Simple Loop

Calculate the cost of a while.

Measuring Cost

```
cont=0
sum =0
while (cont<n):
                    sum+= cont
        cont+=1
print "The sum is: "+ sum
```

Cost (C_i)	Time (T_i)
C_1	1
C_2	1
C_3	n+1
C_4	n
C_5	n
C_6	1

$$Cost = C_1+C_2+(n+1)*C_3+n*C_4+C_5+C_6 = 1+1+n+1+n+n+1 = 3n+4$$

Measuring Cost

Example 3: Double Loop

Calculate the cost of a double while.

Measuring Cost

```
cont_rows=0
sum =0
while (cont_rows<n):
    cont_col=cont_rows
    while (cont_col<n):
        sum+= cont_rows+cont_cols
        cont_cols+=1
    cont_rows+=1
print "The sum is: "+ sum
```

Cost (C_i)	Time (T_i)
C_1	1
C_2	1
C_3	n+1
C_4	n
C_5	n*(n+1)
C_6	n*n
C_7	n*n
C_8	n
C_9	1

$$Cost = \sum_{i=1}^{9} C_i = 3 + (n+1) + 2n + n*(n+1) + 2n^2 = 4 + 4n + 3n^2$$

Measuring Cost

General rules

1. **Loops:** The running time of a loop is at most the running time of the statements inside of that loop times the number of iterations.
2. **Nested Loops:** Running time of a nested loop containing a statement in the inner most loop is the running time of statement multiplied by the product of the sized of all loops.
3. **Consecutive Statements:** Just add the running times of those consecutive statements.
4. **If/Else:** Never more than the running time of the test plus the larger of running times of S1 and S2.

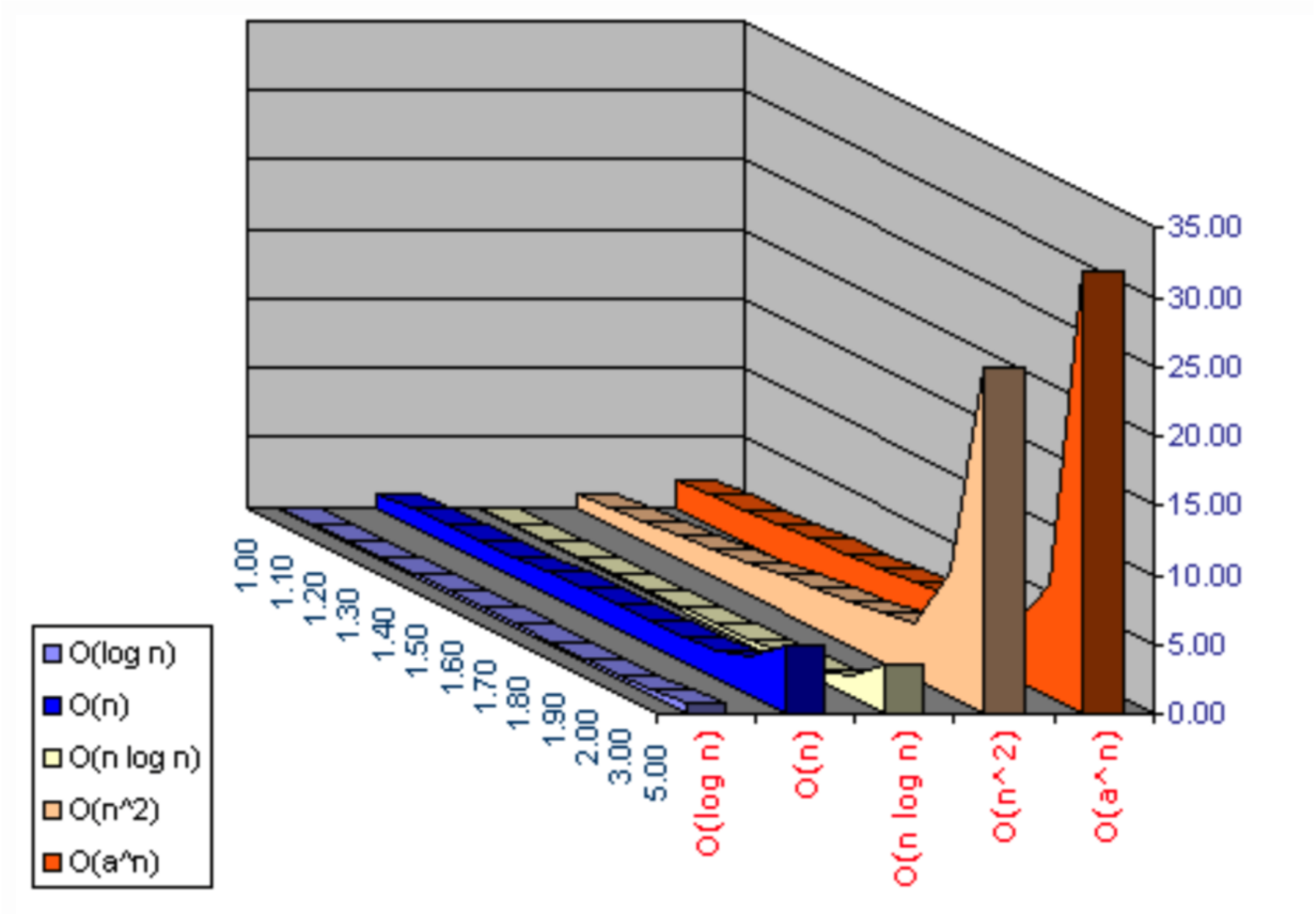
Growth Rates

Growth Rate Definition

As seen, the cost is a mathematical function that represent the time growth of the algorithm execution. This function is called **growth rate**.

Function	**Description**
c	Constant
$logn$	Logarithmic
$log_2 n$	Log-Squared
n	Linear
$n * logn$	Log-linear
n^2	Square
n^3	Cubic
n^m	Exponential

Growth Rates



Big O

Definition Big O

For a given function g(n), we denote by $O(g(n))$ or $\Theta(g(n))$ the set of functions:
$\Theta(g(n)) = O(n) = \{f(n)$: there exist a positive constant c and n_0 such that $0 \leq f(n) \leq c * g(n)\ \forall n \geq n_0\}$

Big O

Example 1: If-structure -based on the example mentioned above-

Calculate $\Theta(g(n))$ of an if-structure:

$$f(n) = C_1 + max(C_2, C_3) = 2$$

Big O

Example 1: If-structure -based on the example mentioned above-

Calculate $\Theta(g(n))$ of an if-structure:

$$f(n) = C_1 + max(C_2, C_3) = 2$$

Solution:

$0 \leq 2 \leq c * g(n)\ \forall n \geq n_0 \Rightarrow c = 2; g(n) = 1 \Rightarrow \Theta(1)$

Big O

Example 2: Simple While -based on the example mentioned above-

Calculate $\Theta(g(n))$ of an if-structure:

$$f(n) = C_1 + C_2 + (n+1) * C_3 + n * C_4 + C_5 + C_6 = 3n + 4$$

Big O

Example 2: Simple While -based on the example mentioned above-

Calculate $\Theta(g(n))$ of an if-structure:

$$f(n) = C_1 + C_2 + (n+1) * C_3 + n * C_4 + C_5 + C_6 = 3n + 4$$

Solution:

$0 \leq 3n + 4 \leq c * g(n)\ \forall n \geq n_0 \Rightarrow 0 \leq 3n + 4 \leq 4n \Rightarrow \Theta(n)$

Big O

Example 3: Nested While -based on the example mentioned above-

Calculate $\Theta(g(n))$ of an if-structure:

$$f(n) = \sum_{i=1}^{9} C_i = 4 + 4n + 3n^2$$

Big O

Example 3: Nested While -based on the example mentioned above-

Calculate $\Theta(g(n))$ of an if-structure:

$$f(n) = \sum_{i=1}^{9} C_i = 4 + 4n + 3n^2$$

Solution:

$$0 \leq 4 + 4n + 3n^2 \leq c * g(n)\ \forall n \geq n_0 \Rightarrow 0 \leq 4 + 4n + 3n^2 \leq 4n^2 \Rightarrow \Theta(n^2)$$

Big O

Strategy to calculate the $\Theta(g(x))$

1. Calculate the cost function (Growth Rate) $\rightarrow$ f(x)
2. Obtain the max exponential of the equation
3. Remove constants $\rightarrow \Theta(g(x))$

What to measure in an algorithm?

Types of Analysis

1. **Worst-Case Analysis:** The maximum amount of time that an algorithm require to solve a problem of size n.
2. **Best-Case Analysis:** The minimum amount of time that an algorithm require to solve a problem of size n.
3. **Average-Case Analysis:** The average amount of time that an algorithm require to solve a problem of size n.

Exercises

1. **Calculate the power between two values without using the "power function" of Python.**
2. **Calculate Fibonacci series given a length.**
3. **Calculate the quotient and the reminder from the division between two integers. Do not use the specific Python operations.**
4. **Calculate the maximum value of an array.**

References

1. Big O cheatsheet: http://bigocheatsheet.com
2. Algorithm Analysis - METU OCW: ocw.metu.edu.tr/pluginfile.php/2548/mod.../0/algorithmAnalysis.ppt
3. Algorithms Analysis- CENG: www.cc.gatech.edu/ bleahy/cs1311/cs1311lecture23wdl.ppt
4. Codeacademy Big-O curse: https://www.codecademy.com/courses/big-o