Measure Algorithms Efficiency

Jordi Planes & Aitor Corchero

Universitat de Lleida

15-02-2016
Index

- Algorithms
- Algorithms Performance (Time vs Space)
- Measuring Cost
- Growth Rates
- Big O
- What to measure in an algorithm?
- Exercises
- References
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.
Is sufficient the algorithm correctness? NO
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)

<table>
<thead>
<tr>
<th>TIME</th>
<th>SPACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions take time</td>
<td>Data structures take space</td>
</tr>
<tr>
<td>Algorithm perform speed?</td>
<td>Data structures to be used?</td>
</tr>
<tr>
<td>What affects its runtime?</td>
<td>How data structure affect the runtime?</td>
</tr>
</tbody>
</table>
# Algorithms Performance (Time vs Space)

<table>
<thead>
<tr>
<th>TIME</th>
<th>SPACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions take time</td>
<td>Data structures take space</td>
</tr>
<tr>
<td>Algorithm perform speed?</td>
<td>Data structures to be used?</td>
</tr>
<tr>
<td>What affects its runtime?</td>
<td>How data structure affect the runtime?</td>
</tr>
</tbody>
</table>

## 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?
Cost Definition

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

\[ \text{metric} = \text{Time} \]

How to measure algorithm time??
Example 1: If structure

Calculate the cost in an if structure
Measuring Cost

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

<table>
<thead>
<tr>
<th>Cost ($C_i$)</th>
<th>Time ($T_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1</td>
</tr>
<tr>
<td>$C_2$</td>
<td>1</td>
</tr>
<tr>
<td>$C_3$</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
Cost = C_1 + \max(C_2, C_3) = 2
\]
Example 2: Simple Loop

Calculate the cost of a while.
Measuring Cost

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

<table>
<thead>
<tr>
<th>Cost ($C_i$)</th>
<th>Time ($T_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1</td>
</tr>
<tr>
<td>$C_2$</td>
<td>1</td>
</tr>
<tr>
<td>$C_3$</td>
<td>n+1</td>
</tr>
<tr>
<td>$C_4$</td>
<td>n</td>
</tr>
<tr>
<td>$C_5$</td>
<td>n</td>
</tr>
<tr>
<td>$C_6$</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
Cost = C_1 + C_2 + (n+1)C_3 + nC_4 + C_5 + C_6 = 1 + 1 + n + 1 + n + n + 1 = 3n + 4
\]
Example 3: Double Loop

Calculate the cost of a double while.
Measuring Cost

```python
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
```

<table>
<thead>
<tr>
<th>Cost ($C_i$)</th>
<th>Time ($T_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>1</td>
</tr>
<tr>
<td>$C_2$</td>
<td>1</td>
</tr>
<tr>
<td>$C_3$</td>
<td>$n+1$</td>
</tr>
<tr>
<td>$C_4$</td>
<td>$n$</td>
</tr>
<tr>
<td>$C_5$</td>
<td>$n*(n+1)$</td>
</tr>
<tr>
<td>$C_6$</td>
<td>$n*n$</td>
</tr>
<tr>
<td>$C_7$</td>
<td>$n*n$</td>
</tr>
<tr>
<td>$C_8$</td>
<td>$n$</td>
</tr>
<tr>
<td>$C_9$</td>
<td>1</td>
</tr>
</tbody>
</table>

Cost = $\sum_{i=1}^{9} C_i = 3 + (n + 1) + 2n + n \times (n + 1) + 2n^2 = 4 + 4n + 3n^2$
## 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.
As seen, the cost is a mathematical function that represents the time growth of the algorithm execution. This function is called **growth rate**.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>Constant</td>
</tr>
<tr>
<td>$\log n$</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>$\log_2 n$</td>
<td>Log-Squared</td>
</tr>
<tr>
<td>$n$</td>
<td>Linear</td>
</tr>
<tr>
<td>$n \times \log n$</td>
<td>Log-linear</td>
</tr>
<tr>
<td>$n^2$</td>
<td>Square</td>
</tr>
<tr>
<td>$n^3$</td>
<td>Cubic</td>
</tr>
<tr>
<td>$n^m$</td>
<td>Exponential</td>
</tr>
</tbody>
</table>
Growth Rates

Figure 1: Algorithms growth rates
Definition Big O

For a given function $g(n)$, we denote by $O(g(n))$ or $\Theta(g(n))$ the set of functions:

$$\Theta(g(n)) = O(n) = \{f(n) : \text{there exist a positive constant } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq c \cdot g(n) \ \forall n \geq n_0\}$$
Example 1: If-structure -based on the example mentioned above-

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

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

$$0 \leq 2 \leq c \cdot g(n) \ \forall n \geq n_0 \Rightarrow c = 2; g(n) = 1 \Rightarrow \Theta(1)$$
Example 2: Simple While -based on the example mentioned above-

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

$$f(n) = C_1 + C_2 + (n + 1) \cdot C_3 + n \cdot C_4 + C_5 + C_6 = 3n + 4$$
Example 2: Simple While -based on the example mentioned above-

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

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

Solution:

$$0 \leq 3n + 4 \leq c \cdot g(n) \quad \forall n \geq n_0 \Rightarrow 0 \leq 3n + 4 \leq 4n \Rightarrow \Theta(n)$$
Example 3: Nested While -based on the example mentioned above-

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

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

$$0 \leq 4 + 4n + 3n^2 \leq c \cdot g(n) \quad \forall n \geq n_0 \Rightarrow 0 \leq 4 + 4n + 3n^2 \leq 4n^2 \Rightarrow \Theta(n^2)$$
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
4. Codeacademy Big-O curse: https://www.codecademy.com/courses/big-o