Big $\mathcal{O}$ notation

Ulises Tirado Zatarain [1]
(ulises.tirado@cimat.mx)

[1] Algorists Group

May, 2016

Outline

1. Introduction
   - The Problem
   - Counting, counting, counting, ...
   - Notation
   - Cases
2. Examples and more rules

A little problem: What kind of algorithm is that?

```
/* uli is an abbreviation for the type:
   unsigned long int.                                */
function do-something-weird(uli x , uli y): uli begin
    uli result ← 0;
    while $x \neq 0$ do
        if $x$ & $1 = 1$ then:
            result ← result + y;
        end
        $x \leftarrow x >> 1$;
        $y \leftarrow y << 1$;
    end
    return result;
end
```

Definitions

- Big $\mathcal{O}$ time/space is the language and metric we use to describe the efficiency of algorithms.
- Imagine the following scenario:
  - You got a file on a hard drive and you need to send it to a friend who lives across the country. You need to get the file to your friend as fast as possible. How should you send it?
  - E-mail
  - FTP, HTTP, SCP, ...
  - Dropbox, Google Drive, Sky Drive ...
  - Airplane
  - What if the file were really large?

Definitions

- The ammount of elemental operations or stored information to solve a problem we know as “Computational Complexity”
- The are two kinds of complexity:
    - Run time (Temporal complexity)
    - Memory (Spatial complexity)
- Big $\mathcal{O}$ notation allow us classify our algorithms in categories based on input size.

Notation $\mathcal{O}(\cdot)$: Categories

- The categories are:
  - $\mathcal{O}(1)$, $\mathcal{O}(n)$, $\mathcal{O}(n^2)$, $\mathcal{O}(n^3)$, $\mathcal{O}(n^4)$, $\mathcal{O}(\log n)$, $\mathcal{O}(n \log n)$, $\mathcal{O}(n!)$, $\mathcal{O}(2^n)$, ...
  - What the hell does this mean? $\mathcal{O}$.o o.$\mathcal{O}$

Definitions and notation

- For example, to send a file to a friend the run time is:
  - By e-mail, FTP, Dropbox is $\mathcal{O}(s)$ where s is the file size.
  - By airplane is $\mathcal{O}(1)$. (But maybe is $\mathrm{O}\left(n^2\right)$ in money)
  - Why?
- Rule: Drop the constants
  - What?
  - Why?
  - Example: seems that to send by e-mail is indeed at least $2s$ in time where s is the file size.

Best, Worst and Average/Expected cases

- Think about search an element in an unsorted simply linked list:
    - Best case: the element is at the start of the list. (Maybe you think this is $\mathcal{O}(1)$).
    - Worst case: the element is at the end of the list. (Maybe you think this is $\mathcal{O}(n)$).
    - Average/Expected case: the element is in any position at the list. (What do you think about this?)

Print elements in arrays A and B

$\mathcal{O}(a+b)$: where a and b are the size of array A and B respectively.

```
for each $a$ in $A$ do
    print(a);
end
for each $b$ in $B$ do
    print(b);
end
```

$\mathcal{O}(ab)$: where a and b are the size of array A and B respectively.

```
for each $a$ in $A$ do
    for each $b$ in $B$ do
        print($a$,$b$);
    end
end
```

Print elements in arrays A

$\mathcal{O}\left(n^2\right)$: where n is the size of array A.

```
for $i = 0$ to $n-1$ do
    for $j = i$ to $n-1$ do
        print(a[j]);
    end
end
```

$$
\begin{array}{l}
a[0], a[1], a[2], \cdots, a[n-1] \\
a[1], a[2], \cdots, a[n-1] \\
a[2], \cdots, a[n-1] \\
\vdots \\
a[n-1]
\end{array}
$$

What do following code? What are their run time?

a and b are integers.

```
function something(a,b) begin
    if $b \leqslant 0$ then:
        return $-1$;
    end
    d = a / b;
    return $a - d * b$;
end
```

a and b are integers.

```
function something(a,b) begin
    if $b \leqslant 0$ then:
        return $-1$;
    end
    $counter \leftarrow 0$;
    $sum \leftarrow 0$;
    while $sum \leqslant a$ do
        $sum \leftarrow sum + b$;
        $counter \leftarrow counter + 1$;
    end
    return $counter$;
end
```

What does following code? What is their run time?

n is a integer.

```
function something(n) begin
    $x \leftarrow 0$;
    while $x * x < n$ do
        $x \leftarrow x + 1$;
    end
    return $x$;
end
```

What does following code? What is their run time?

n is a integer.

```
function something(n) begin
    $x \leftarrow 0$;
    while $x * x < n$ do
        $x \leftarrow x + 1$;
    end
    return $x$;
end
```

What does following code? What is their run time?

```
/* Take the assumption that you can concatenate two
   strings or string and character with operator $+$  */
function something(char $*s$,char $*p=$“”) begin
    if $*s=0$ then:
        print(p);
    else:
        char $*r \leftarrow s$;
        while $*r \neq 0$ do
            char $*q \leftarrow$ substring(s,r)+substring($r+1$);
            something(q,$p+*r$);
            $r \leftarrow r+1$;
        end
    end
end
```

Function to get substring used in something function

```
function substring(char $*s$,char $*r = 0$) begin
    if $r \neq 0$ then:
        char $*q \leftarrow$ new char $[r - s + 1]$; int $k \leftarrow 0$;
        while $s < r$ do
            $q[k] \leftarrow *s$; $s \leftarrow s + 1$; $k \leftarrow k + 1$;
        end
        $q[k] \leftarrow 0$;
        return $q$;
    else:
        $r \leftarrow s$;
        while $*r \neq 0$ do $r \leftarrow r + 1$ ;
        return substring($s$,$r$);
    end
end
```

References I

- Gayle Laakmann - Cracking the Coding Interview
- Robert Sedgewick - Algorithms C++
- Thomas H. Cormen - Introduction to Algorithms
- Donald E. Knuth - The Art of Computer Programming
- Wikipedia
- Quora