Latex-Analiza Complexităţii

Constantin Ionuţ Andrei

May 25, 2017

Instrucţiuni simple complexitate O(1)
a = a + 1
a > 0
a - 1

Instrucţiunea if

```
if conditie {                    complexitate q+1
   .........
   ......... } => q
} else {
   .........
   ......... p                   complexitate p+1
}
```

Instrucţiunea switch

```
Switch   x   {               Complexitate fara break=
     case 1: ......          nr case + complexitate true
     case 2: ......             sau complexitate default
     case 3 : .....          complexitate cu break
     default : ....             aceeasi si de mai sus daca
                                nu sunt multiple cazuri
}
     switch   x   {
   case 2: ..... break
   case 2: .....
   default: .....
}
```

Instrucţiunea while...

conditie

while $(x \leq n)$ {   Complexitate: nr pasi x $(q+1)$

....

.... q

....

}

Instrucţiunea do...while

do {

......

...... q   Complexitate

......

} while(conditie)

$x = 0;$

do { $(nr\ pasi)\ (q+1)$

$x = x + 1;$

} while $(x \leq 3);$

x=1   x=4

comp true   comp false

$x = 2$

com true

$x = 3$

comp true

Instrucţiunea for

for $(start, stop, increment)$ {
....
.... q

} Complexitate: $1 + nr\ pasi$ x $(q + 2)$

---

for $(i = 0; i \leq 100; i = i + 2)$ {
if $(i\%10 == 0)$ { nr pasi=51
print ....
}
} $1 + 51(+)$

1+11(1+1)+40x1

Ordinul complexitatii $= 1 + 22 + 40 = 63$ complexitate

-O(1)

-O($log\, n$) logaritmica $1 + (\frac{n}{2} + 1)(+)$

-O(n) liniara

-O(n^2) patratic $1 + \frac{4n}{10}$x1 $+ (\frac{n}{10} + 1)$x2

-O(n^3) cubic

-O(x^n) exponential Complexitatea se calculeaza pentru
n $- > \infty$
- input max

O(n) "big OH" -limita superioara - upper bound

$\mathrm{O}(g(n)) = \{\ f(n) : E_c,\ \mathrm{n}_0\ \ a.i. f(\mathrm{n}) \leq g(n),\quad n \geq n_0\ \}$

$\mathrm{f}(n) = 5n^2 + 2n + 1\ =\ \mathrm{O}(n^2)$ C$=5 + 2 + 1$
$\mathrm{n}_0 = 1$

$\mathrm{f}(n) \leq 8n^2$

$\boxed{\mathrm{f}(n) \leq Cn^2\quad ,\quad \mathrm{n}\ \geq\ \ n_0}$

$\Omega - Omega\qquad - limita\ inferioara\quad lower\ bound$

$\Omega(\mathrm{g(n)})\ = \{f(\mathrm{n}):\ \ \exists\ C,\ \ n_0\ \ a.i.\ \ g(\mathrm{n})$
$\geq f(\mathrm{n}),\ \ n \leq n_0\}$

$\mathrm{f}(n) = 5n^2 + 2n + 1 = \Omega(n^2)$

$\mathrm{C} = 5\ \mathrm{n}_0$ $5\mathrm{n}^2 + 2n + 1 \geq 5n^2\ \ pt\ \ n \geq 0$

$$\theta \ - \ Theta \ \ - growth \ \ \ of \ \ function \ \ f$$

$$\theta \text{ (g(n))} \ = \ \{ \text{ f(n):} \ \ \exists \ C_1, \ \ C_2, \ \ n_0 \qquad \text{a.i.}$$

$$\text{C}_1\text{g(n)} \leq \ \text{f(n)} \leq \ \ C_2\text{g(n)}, \ \ n \geq n_0\}$$

$$f(n) \ = \ \ 5n^2 + 2n + 1 \ = \ \theta(\text{n}^2)$$

$$\text{C}_1 = 5 \quad C_2 = 8 \quad n_0 = 1$$

$$5n^2 \leq \ \ 5n^2 + 2n + 1 \leq \ \ 8n^2 \text{ pt n} \geq 1$$

procedure bubbleSort(A)

n = length(A)

repeat

swapped = false O(1)

for i = 1 to $n-1$ do

if $A[i-1] > A[i]$ then

swap ($A[i{-}1], A[i]$) O(3)

swapped = true O(1)

until not swapped

$[1 + (n-1)(5+2) + 1] \ \ (n-1)$

$= \ 7(n-1)^2 + 2n - 2 \ = \ 7(n^2 - 2n + 1) + 2n + 2$

$o_1 \ \ o_2 \ \ \ldots\ldots\ldots \ \ o_{n-1} \qquad\qquad o_n = \boxed{7n^2 - 12n + 3}$

$(n-1)$ swaps required in worst case

1 repeat	2 repeat	3 repeat
I 4 3 2 1	I 3 2 1 4	I 2 1 3 4
II 3 4 2 1	II 2 3 1 4	II 1 2 3 4
III 3 2 4 1	III 2 1 3 4	
IV 3 2 1 4		

O(n) = $7n^2 =\ O(n^2)$
Ω(n) = 6n^2 pt n$\approx$ 6+$\sqrt{33}$ $\approx$ $13,75$ =On^2)

$$7n^2 - 12n + 3 = 7n^2$$

$$-12n + 3 = 0$$

$$3 = 12n => n = \frac{3}{12} = \frac{1}{4}$$

$$7(\frac{1}{4})^2 - 12\frac{1}{4} + 3 = 7\frac{1}{16} - 3 + 3$$

$$= \frac{7}{16}$$

$$\boxed{7n^2 - 12n + 3 = 5n^2}$$

$$7n^2 - 12n + 3 = 6n^2$$

$$\boxed{2n^2 - 12n + 3 = 0}$$

$$\Delta = b^2 - 4 * a * c$$

$$\Delta = 144 - 12$$

$$\Delta = 132$$

$$x_1 = \frac{-b+\sqrt{\Delta}}{2*a} = \frac{12+\sqrt{1}32}{2} = 6 + \sqrt{3}3 \simeq 6 + 5,75 = 13,75$$

O(n) $\quad 6 * n^2 \leq f(n) \leq 7 * n^2$
O(n) $= n^2 \quad 7 * n^2 - 12 * n + 3 = 5 * n^2$

$$2 * n^2 - 12 * n + 3 = 0$$
$$\Delta = 144 - 24 = 120$$
$$x_2 = \frac{12+\sqrt{1}20}{4} = 3 + \frac{\sqrt{1}20}{4}$$

recursive

Factorial

```
f(n) {
    if n=0 return 1          O(1)
    else return n*f(n-1)
}
```

T(n) = T(n-1) + 3 $\quad n \geq 0$

=T(n-2) + 6

=T(n-3) + 9

=T(n-k) + 3*k

T(0) $\Rightarrow$ n-k=0 $\Rightarrow$ k=n

T(n) = T(0) + $3 * n = 1 + 3 * n$ = O(n)

Fibonaci

```
Fib(n) {
    if n ≤ 0 return 1
    else return Fib(n-1) + Fib(n-2)          O(1)
}
```

$T(n) = T(n-1) + T(n-2) + 4$
$T(0) = T(1) = 1$
....
Presupunem T(n-1) = T(n-2) dar T(n-1) $\geq$ T(n-2) approximare in jos c = 4 constantă

$$
\begin{aligned}
T(n) &= 2 * T(n-2) + 4 \\
&= 2 * (2 * T * (n-4) + c) + c \\
&= 4 * T(n-4) + 3 * c) \\
&= 8 * T(n-6) + 7 * c) \\
&= 16 * T(n-8) + 15 * c) \\
&= 2^k * T * (n-k) + (2^k - 1) * c
\end{aligned}
$$

$T(0) => n - 2 * k = 0 => k = n/2$

$$
\begin{aligned}
T(n) &= 2^{n/2} * T(0) + (2^{n/2} - 1) * c \\
&= 2^{n/2} * (1 + c) - c
\end{aligned}
$$

$T(n) = 2^{n/2} \qquad \Omega(2^{n/2})$

Presupunem T(n-1) = T(n-2) dar T(n-1) $\leq$ T(n-2) deci maximizam functia

$$
\begin{aligned}
T(n) &= 2 * T * (n-1) + c \qquad \text{c = 4} \\
&= 2 * (2 * T(n-2) + c) + c \\
&= 4 * T(n-2) + 3 * c \\
&= 8 * T(n-3) + 7 * c \\
&= 2^k * T(n-k) + (2^k - 1) * c
\end{aligned}
$$

$T(0) => k = n => T(n) * 2^n * T(0) + (2^n - 1) * c$

$$
\begin{aligned}
T(n) &= 2^n * T(0) + 2^n * c - c \\
&= 2^n * (T(0) + c) - c \\
&= 2^n * (1 + c) - c
\end{aligned}
$$

$O(n) = 2^n$

Fibo non-recursive

```
int fibo(n) {
   if(n≤1) return 1
   rez = 1                        O(n)
   rezPre = 1
   for i=2,n
        temp = rez
        rez = rez + rezPre
        rezPre = temp
   return rez
}
```

Figure 1: Graph 1

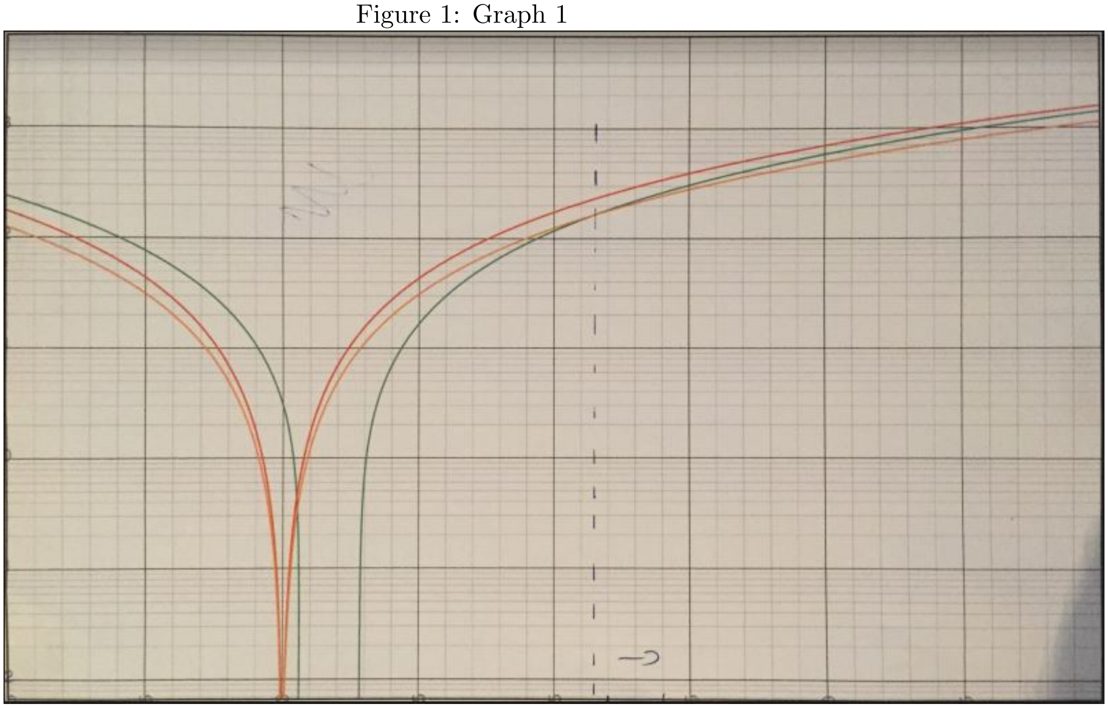

Figure 2: Graph 2