Programación 3

Notación $\mathcal{O}$

Alejandro Mujica
Jesús Pérez

Semestre A2017

Definición

Notación $\mathcal{O}$

Sean dos funciones $T(n)$ y $f(n)$. Entonces se dice que $T(n)$ es $\mathcal{O}(f(n))$, denotado como $T(n) = \mathcal{O}(f(n))$, si y sólo si $\exists c, n_1 | T(n) \leq cf(n), n \geq n_1$.

Otras notaciones

Notación o

Sean dos funciones $T(n)$ y $f(n)$. Entonces se dice que $T(n)$ es $o(f(n))$, denotado como $T(n) = o(f(n))$, si y sólo si $\exists c, n_1 | T(n) < cf(n), n \geq n_1$.

Notación Ω

Sean dos funciones $T(n)$ y $f(n)$. Entonces se dice que $T(n)$ es $\Omega(f(n))$, denotado como $T(n) = \Omega(f(n))$, si y sólo si $\exists c, n_1 | T(n) \geq cf(n), n \geq n_1$.

Notación ω

Sean dos funciones $T(n)$ y $f(n)$. Entonces se dice que $T(n)$ es $\omega(f(n))$, denotado como $T(n) = \omega(f(n))$, si y sólo si $\exists c, n_1 | T(n) > cf(n), n \geq n_1$.

Notación Θ

Sean dos funciones $T(n)$ y $f(n)$. Entonces se dice que $T(n)$ es $\Theta(f(n))$, denotado como $T(n) = \Theta(f(n))$, si y sólo si $T(n) = \mathcal{O}(f(n))$ y $T(n) = \Omega(f(n))$

Álgebra de $\mathcal{O}$

- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(max(f(n), g(n)))$
- $c\mathcal{O}(f(n)) = \mathcal{O}(f(n))$
- $\mathcal{O}(\mathcal{O}(f(n))) = \mathcal{O}(f(n))$
- $\mathcal{O}(f(n))\mathcal{O}(g(n)) = \mathcal{O}(f(n)g(n))$
- $T(n) = n^k + n^{k-1} + \cdots + n + c = \mathcal{O}(n^k)$
- $log^k n = \mathcal{O}(n)$

Bloques clásicos de programación

Instrucciones secuenciales

Dos instrucciones secuenciales con tiempos $\mathcal{O}(f_1(n))$ y $\mathcal{O}(f_2(n))$ respectivamente, toman la suma $\mathcal{O}(f_1(n)) + \mathcal{O}(f_2(n))$.

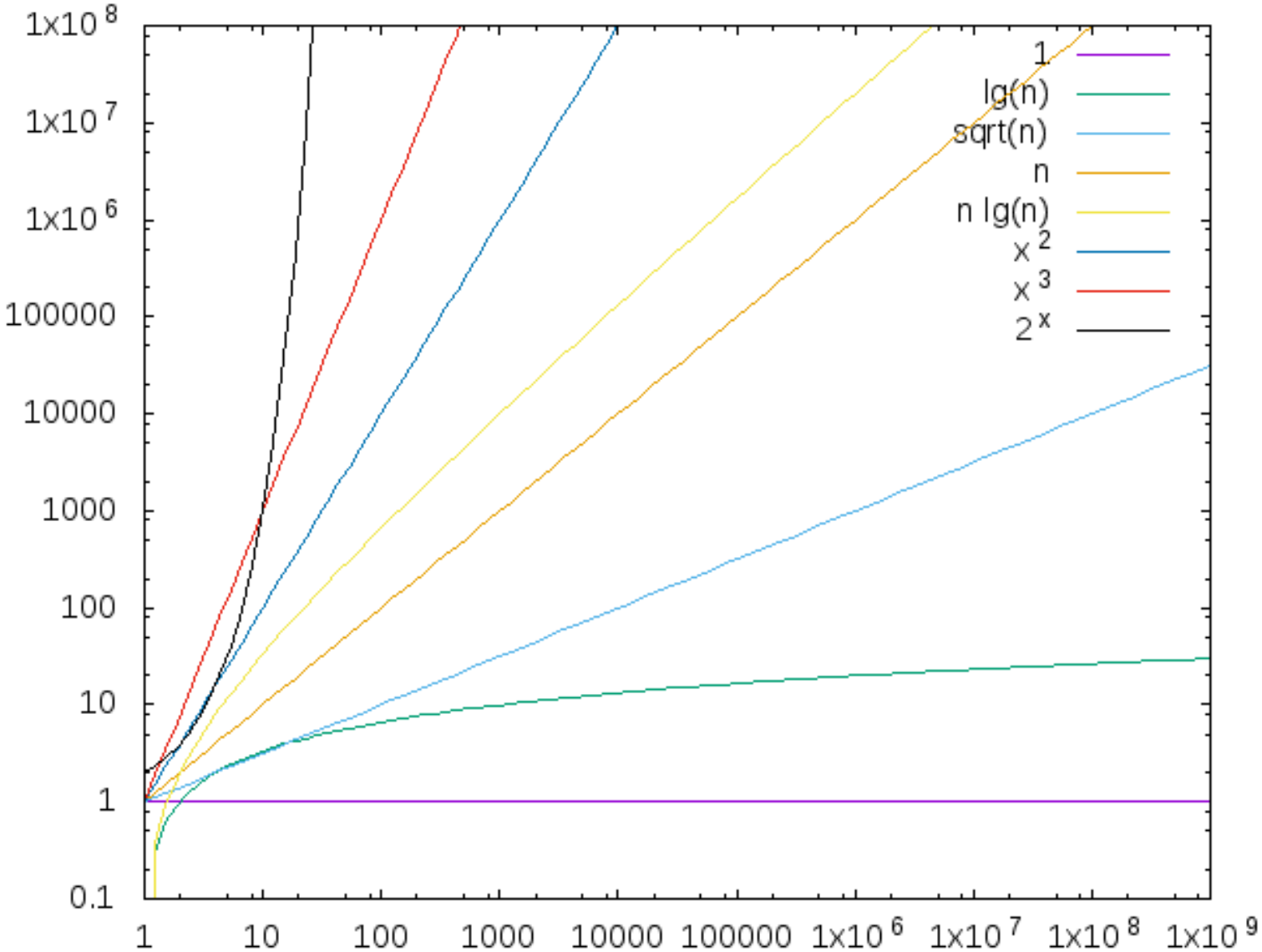
Instrucciones de decisión

if $(\mathcal{O}(f_1(n)))$
    $\mathcal{O}(f_2(n))$
else
    $\mathcal{O}(f_3(n))$
Toma $\mathcal{O}(f_1(n)) + max(\mathcal{O}(f_2(n)), \mathcal{O}(f_3(n)))$.

Instrucciones de repetición

for$(\mathcal{O}(f_1(n)))$
    $\mathcal{O}(f_2(n))$
Toma $\mathcal{O}(f_1(n))\mathcal{O}(f_2(n))$.

Conjunto de funciones descriptoras de algoritmos



Análisis de la búsqueda del menor elemento

```
size_t search_min(int * a, size_t n)
{
  size_t min = 0;

  for (size_t i = 1; i < n; ++i)
    if (a[i] < a[min])
      min = i;

  return min;
}
```

Ordenamiento por inserción

```
void insertion_sort(int * a, size_t n)
{
  for (size_t i = 1; i < n; ++i)
    {
      int tmp = a[i];

      for (size_t j = i; j > 0 and tmp < a[j - 1]; --j)
        a[j] = a[j - 1];

      a[j] = tmp;
    }
}
```