# Introducción a la Computación (Matemática)

Primer Cuatrimestre de 2019

Algoritmos de Ordenamiento

# Complejidad temporal

$T(n)$: Tiempo de ejecución (o complejidad temporal) de un programa; medido en cantidad de operaciones en función del tamaño de la entrada.

- **Peor caso**: $T(n)$ es una cota superior del tiempo de ejecución para entradas arbitrarias de tamaño $n$.

Orden de complejidad temporal: $T(n) \in O(f(n))$ si existen constantes enteras positivas $c$ y $n_0$ tales que para $n \geq n_0$, $T(n) \leq c \cdot f(n)$.

- **Ejemplo:** $T(n) = 3n^3 + 2n^2 \in O(n^3)$, dado que con $n_0 = 0$ y $c = 5$, vale que para $n \geq 0$, $T(n) \leq 5 \cdot n^3$.

En general, nos interesa buscar algoritmos con el menor orden posible, para poder procesar entradas arbitrariamente grandes.

# Cálculo de órdenes de complejidad

**Instrucciones minimales:** lectura/escritura de una variable o de una posición en un arreglo, longitud de un arreglo, operaciones simples de tipos básicos. Orden constante: $O(1)$.

**Secuencialización:** Si $S_1$ y $S_2$ tienen $O(f)$ y $O(g)$, resp., entonces $S_1; S_2$ tiene $O(f) + O(g) = O(\text{máx}(f, g))$.

**Condicional:** Si $B$, $S_1$ y $S_2$ tienen $O(f)$, $O(g)$ y $O(h)$, if $(B)$ $S_1$ else $S_2$ tiene $O(f) + O(\text{máx}(g, h)) = O(\text{máx}(f, g, h))$.

**Ciclo:** Si $B$ y $S$ tienen $O(f)$ y $O(g)$, y se ejecutan $O(h)$ veces, entonces while $(B)$ $S$ tiene $O(f + g) \cdot O(h) = O((f + g) \cdot h)$.

# Algoritmo de búsqueda lineal

Encabezado: $Buscar : x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \rightarrow está \in \mathbb{B} \times pos \in \mathbb{Z}$

Precondición: $\{A = A_0 \wedge x = x_0\}$

Poscondición: $\{(está = true \wedge 0 \leq pos < |A_0| \wedge A_0[pos] = x_0) \vee$
$(está = false \wedge (\forall i)(0 \leq i < |A_0| \Rightarrow A_0[i] \neq x_0))\}$

$está \leftarrow false$ $O(1)$

$pos \leftarrow -1$ $O(1)$

$j \leftarrow 0$ $O(1)$

while ($j < |A|$) { $O(1)$ while: $O(|A|)$ iteraciones

  if ($A[j] = x$) { $O(1)$

    $está \leftarrow true$ $O(1)$

    $pos \leftarrow j$ $O(1)$

  }

  $j \leftarrow j + 1$ $O(1)$

}

Complejidad temporal: $O(1 + 1 + 1 + 1 + |A| \cdot (1 + 1 + 1 + 1 + 1))$

$= O(1 + |A| \cdot 1) = O(|A|)$

# Algoritmo de búsqueda binaria

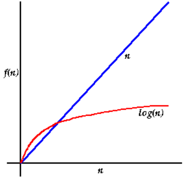
Encabezado: $Buscar : x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \rightarrow está \in \mathbb{B} \times pos \in \mathbb{Z}$
Precondición: $\{A = A_0 \wedge x = x_0 \;\wedge \text{Creciente}(A_0)\}$
Poscondición: $\{(está = true \wedge 0 \leq pos < |A_0| \wedge A_0[pos] = x_0) \vee$
$(está = false \wedge (\forall i)(0 \leq i < |A_0| \Rightarrow A_0[i] \neq x_0))\}$

$(está, pos) \leftarrow (false, -1)$
$(izq, der) \leftarrow (0, |A| - 1)$
while $(izq < der)$ $\{$ O(1) while: $O(\log_2 |A|)$ iteraciones
  $med \leftarrow (izq + der)$ div 2 $O(1)$
  if $(A[med] < x)$ $\{$ $O(1)$
    $izq \leftarrow med + 1$ $O(1)$
  $\}$ else $\{$
    $der \leftarrow med$ $O(1)$
  $\}$
$\}$
if $(x = A[izq])$ $\{$ $O(1)$
  $(está, pos) \leftarrow (true, izq)$ $O(1)$
$\}$ Complejidad temporal: $O(\log |A|)$

# Problema de ordenamiento

Problema de búsqueda:

- $O(n)$ para listas arbitrarias.
- $O(\log n)$ para listas ordenadas.




Ordenar una lista es un problema de mucha importancia en la práctica.

- Ej: ¿De qué sirve una guía de teléfonos desordenada?

Encabezado: $\mathit{Ordenar} : A \in \mathbb{Z}[\,] \to \emptyset$
Precondición: $\{A = A_0\}$
Poscondición: $\{\text{Permutación}(A, A_0) \land \text{Creciente}(A)\}$

¿Ideas para resolver este problema?

# Problema de ordenamiento

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 59 | 7 | 388 | 41 | 2 | 280 | 50 | 123 |

# Selection sort

Para cada $i$ entre 0 y $|A|-1$, buscar el menor elemento en $A[i\,..\,|A|-1]$ e intercambiarlo con $A[i]$.

$i \leftarrow 0$ $\quad O(1)$
while $(i < |A|)$ { $\quad$ O(1) $\qquad$ while: $O(|A|)$ iteraciones
$\quad$ $posmin \leftarrow i$ $\quad O(1)$
$\quad$ $j \leftarrow i + 1$ $\quad O(1)$
$\quad$ while $(j < |A|)$ { $\quad$ O(1) $\qquad$ while: $O(|A|)$ iteraciones
$\quad\quad$ if $(A[j] < A[posmin])$ { $\quad O(1)$
$\quad\quad\quad$ $posmin \leftarrow j$ $\quad O(1)$
$\quad\quad$ }
$\quad\quad$ $j \leftarrow j + 1$ $\quad O(1)$
$\quad$ }
$\quad$ swap$(A, i, posmin)$ $\quad O(1)$
$\quad$ $i \leftarrow i + 1$ $\quad O(1)$
}

Complejidad temporal:

$$O\big(1 + 1 + |A| \cdot (1 + 1 + 1 + 1 + |A| \cdot (1 + 1 + \text{máx}(1, 0) + 1) + 1 + 1)\big) =$$
$$O\big(1 + |A| \cdot (1 + |A| \cdot (1))\big) = O\big(1 + |A| + |A|^2\big) = O(|A|^2)$$

# Selection sort

Para cada $i$ entre 0 y $|A|-1$, buscar el menor elemento en $A[i\mathinner{..}|A|-1]$ e intercambiarlo con $A[i]$.

$$\text{Inv} \equiv \text{Permutación}(A, A_0) \wedge 0 \le i \le |A| \wedge \text{Creciente}(A, 0, i-1) \wedge \big(0 < i < |A| \Rightarrow \text{Máx}(A, 0, i-1) \le \text{Mín}(A, i, |A|-1)\big)$$

```
i ← 0
while (i < |A|) {
    posmin ← i
    j ← i + 1
    while (j < |A|) {
        if (A[j] < A[posmin]) {
            posmin ← j
        }
        j ← j + 1
    }
    swap(A, i, posmin)
    i ← i + 1
}
```

# Insertion sort

Para cada $i$ entre 0 y $|A| - 1$, mover el elemento $A[i]$ a su posición correcta en $A[0 \mathinner{..} i]$ (así, $A[0 \mathinner{..} i]$ queda ordenado).

$i \leftarrow 0$ $\quad O(1)$
while $(i < |A|)$ { $\quad$ O(1) $\qquad$ while: $O(|A|)$ iteraciones
$\quad$ $j \leftarrow i$ $\quad O(1)$
$\quad$ while $(j > 0 \wedge A[j-1] > A[j])$ { $\quad$ O(1) $\qquad$ while: $O(|A|)$ iters
$\quad\quad$ swap$(A, j-1, j)$ $\quad O(1)$
$\quad\quad$ $j \leftarrow j - 1$ $\quad O(1)$
$\quad$ }
$\quad$ $i \leftarrow i + 1$ $\quad O(1)$
}

Complejidad temporal: $O(|A|^2)$

# Bubble sort

Comparar cada par de elementos adyacentes en $A$, e invertirlos si están en orden incorrecto. Repetir $|A|$ veces.

$i \leftarrow 0$ $\quad O(1)$
while $(i < |A|)$ { $\quad$ O(1) $\qquad$ while: $O(|A|)$ iteraciones
$\quad j \leftarrow 0$ $\quad O(1)$
$\quad$ while $(j < |A| - 1)$ { $\quad$ O(1) $\qquad$ while: $O(|A|)$ iteraciones
$\quad\quad$ if $(A[j] > A[j+1])$ { $\quad O(1)$
$\quad\quad\quad$ swap$(A, j, j+1)$ $\quad O(1)$
$\quad\quad$ }
$\quad\quad j \leftarrow j + 1$ $\quad O(1)$
$\quad$ }
$\quad i \leftarrow i + 1$ $\quad O(1)$
}

Complejidad temporal: $O(|A|^2)$

# Complejidad y Ordenamiento

Bibliografía:


- Aho, Hopcroft & Ullman, “Estructuras de Datos y Algoritmos”, Addison-Wesley, 1988.
- Balcazar, “Programación metódica”, McGraw-Hill, 1993.


Demos y otras yerbas:

- http://www.sorting-algorithms.com/
- http://www.youtube.com/watch?v=MtcrEhrt_K0
- http://www.youtube.com/watch?v=INHF_5RIxTE

# Repaso de la clase de hoy

- Algoritmos de ordenamiento de listas:
  - Selection sort. $O(n^2)$
  - Insertion sort. $O(n^2)$
  - Bubble sort. $O(n^2)$

**Próximos temas**

- Recursión algorítmica.
- Divide and Conquer. Torres de Hanoi.
- Merge sort. $O(n \log n)$
- Cálculo de complejidad algorítmica para funciones recursivas.