El Misterio de EDD

- Al profesor Mario le llega un correo de la universidad preguntando si el alumno Misterio Zallen está incluido en la sección 2 del curso
- Pero la universidad aún no termina el nombramiento de Mario, por lo que no puede ver su curso en Canvas (basado en hechos reales)
- Mario acude al profesor Yadran…

El Misterio de EDD

¿ Zallen Misterio ∈

Alen Misterio
Misterio Misterio
Gonzalópez D
Zalen B
Gonzalópez J
Gazali Misterio
Misterio Yadran
Allen Javier
Zeta Hache
Ararán Jota
Alenn Cristóbal
…

?

El Misterio de EDD

¿ Zallen Misterio ∈

Aaa Yadran
Aab Cristóbal
Aac Yadran
Aca Javier
Acb Javier
Acb Yadran
Acb Cristóbal
Acc Yadran
Acd Javier
Ace Yadran
Ace Yadran
…

?

El Misterio de EDD

¿ 245 ∈

1
2
3
56
57
64
68
99
124
125
126
…

?

Secuencias ordenadas

Una secuencia de números $x_1, \cdots, x_n$ se dice **ordenada** (no decrecientemente) si cumple que $x_1 \leq \cdots \leq x_n$

¿Qué es entonces **ordenar** una secuencia de números?

El Misterio de EDD

¿ Zallen Misterio ∈

Alen Misterio
Misterio Misterio
Gonzalópez D
Zalen B
Gonzalópez J
Gazali Misterio
Misterio Yadran
Allen Javier
Zeta Hache
Ararán Jota
Alenn Cristóbal
…

?

El algoritmo de ordenación del profesor Yadran

1. En la lista original, encontrar el menor valor
2. Borrarlo
3. Escribirlo al final (en el primer espacio disponible) de la lista nueva
4. Si quedan valores en la lista original, entonces volver al paso 1

¿Es correcto el algoritmo del profesor Yadran?

¿Qué quiere decir que un algoritmo sea correcto?

Para nosotros (en este curso) dos propiedades:

- termina en una cantidad finita de pasos
- cumple su propósito, es decir (en este caso), **ordena** los datos

Recordatorio

Demostración **por inducción**:

1.- **Caso base**. Se cumple para $n=1$.

2.- **Paso inductivo**. Si se cumple para $n=k$ (*hipótesis inductiva*), entonces se cumple para $n=k+1$.

Ahora … a trabajar ustedes

Demuestra que el algoritmo anterior es correcto:

- termina en una cantidad finita de pasos
- cumple su propósito, es decir, **ordena** los datos

Termina en una cantidad finita de pasos

- La cantidad de valores a ordenar es finita, digamos *n.*

- En cada vuelta, borramos un valor de la lista original y lo escribimos en la lista nueva.

- Después de *n* vueltas, todos los valores en la lista original fueron borrados. Debido al paso 4, el algoritmo termina.

Cumple su propósito: ordena los datos

Demostración **por inducción**:

- **Caso base**. El primer valor borrado en la lista original y escrito en la lista nueva es el menor de todos (criterio de selección) y está ordenado (único valor en la lista nueva): ✓

- **Hipótesis inductiva**. Los *k* primeros valores borrados en la lista original y escritos en la lista nueva son los *k* valores más chicos y están ordenados

- **Por demostrar** (usando la hipótesis inductiva): *k*+1 **…**

Por demostrar, usando la hipótesis inductiva

Los k+1 primeros valores borrados en la lista original y escritos en la lista nueva son los k+1 valores más chicos y están ordenados:

- los primeros k valores en la lista nueva son los k más chicos (por hipótesis inductiva) y están borrados de la lista original (por paso 2); el siguiente valor que pasa a la lista nueva es el menor de los restantes (por criterio de selección) → el k+1 más chico

- los primeros k números en la hoja nueva están ordenados (por hipótesis inductiva); el siguiente número que se escribe al final de la hoja nueva no es menor que ninguno de los k números que ya están en la hoja nueva (por criterio de selección) → queda ordenado

¡Cumple su propósito!

Demostración **por inducción**:

1.- **Caso base**. Se cumple para $n=1$.

2.- **Paso inductivo**. Si se cumple para $n=k$ (*hipótesis inductiva*), entonces se cumple para $n=k+1$.

El algoritmo *selection sort*

Para la secuencia inicial de datos, A:

1. Definir una secuencia ordenada, B, inicialmente vacía
2. Buscar el menor dato x en A
3. Sacar x de A e insertarlo al final de B
4. Si quedan elementos en A, volver a 2.

¿Cuál es la complejidad de *selection sort*?

Raciocinio para determinar la complejidad de *selection sort*

Buscar el menor dato en A significa revisar A entero: $O(n)$

Este proceso se hace una vez por cada dato: n veces

La complejidad es entonces $n \cdot O(n) = O(n^2)$

Otra forma de calcular la complejidad de *selection sort*

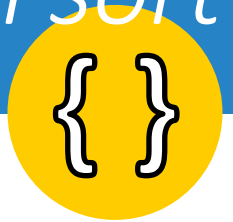
También se puede hacer de manera explícita:

Buscar el mínimo cuesta $n-1$, y el siguiente $n-2$, y así:

$$T(n) = \sum_{i=1}^{n-1} i = \frac{n^2 - n}{2}$$

$$T(n) \in O(n^2)$$

Complejidad de memoria de *selection sort*



selection sort se puede hacer en un solo **arreglo**, ya que $|A| + |B| = n$

Eso significa que no necesita memoria adicional … o, más precisamente, necesita $O(1)$ memoria adicional

Los algoritmos que tienen esta propiedad se conocen como *in place*