

La pequeña ovejería

- Don Juan tiene 600 ovejas
- Cada oveja tiene asignado un número único
- El rebaño está dividido en 4 lotes de 150 ovejas c/u
- Por cada lote hay una hoja con sus números
- Los números de un lote están en cualquier orden

TAIWAN

4160 T
4022 T
3117 m/T
3058 T
5193 S
2103 R
288 B
5/n T
3150 T
4254 m
5008 m/S
246 m

flaca
super flaca

3195 T
2199 M
6175 T
2181 T
293 T
274 T
5088 M
2063 T
4120 A
2021 T

3094 T
2055 m
229 T
3003 T
6004 T
*2140 m
263 m
*2209 T
296 m
3091 m
2098 m/T
2125 B
4195 T

269 m/n
625 T
251 R
6025 S
3108 T
5/n M
3197 M
3300 T
4163 R
6155 T

6106 m
6191 T
2006 m
267 m/n
2084 m/T
3031 T
2039 T
6063 m/n
5/n m/T
248 m
283 m
3137 m
2078 T

4058 M
2167 T
3032 T
6181 m/n
253 M
4114 T
6169 PARDA
6121 PARDA
4021 T
5209 PARDA

252 m
4083 m
2009 m
3014 T
2036 T
2241 m
6247 T
5/n T
2117 m
4177 m/T
268 m

6222 T
2109 m
3180 m
6197 parda
237 B
4206 T
6055 p
5/n p
5154 p/n
3029 m/T

La pequeña ovejería



Don Juan suele encontrar una oveja separada de su lote

¿Cómo puede saber fácilmente a que lote pertenece?

Definición



Una secuencia de números x_1, \dots, x_n se dice **ordenada** si cumple que $x_1 \leq \dots \leq x_n$

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

El algoritmo de Don Juan

1. De la hoja original, tomar el número más pequeño
2. Tacharlo en la hoja original
3. Escribirlo al final de la hoja nueva
4. Si quedan números en la hoja original, volver al 1.

¿Es correcto el algoritmo de Don Juan?

Definición

Un algoritmo se dice **correcto** si cumple que:

- Termina en una cantidad finita de pasos
- Cumple su propósito

El algoritmo de Don Juan



Demuestre que el algoritmo de Don Juan es correcto

Es decir

- Termina en una cantidad finita de pasos
- Cumple su propósito: **ordenar** los datos

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.

Selection Sort



¿Cuál es la complejidad de este algoritmo?

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 $n \cdot O(n) = O(n^2)$

Complejidad de Selection Sort

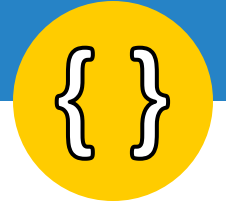
También se puede hacer de manera explícita:

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

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

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

Selection Sort



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

Eso significa que no necesita nada de memoria adicional

Los algoritmos que hacen esto se conocen como *in place*

La pequeña ovejería



Don Juan quiere cambiar 5 ovejas del lote A al lote B

Necesita actualizar el cambio en ambas hojas

¿Cómo lo hace para no tener que volver a ordenar todo?

La pequeña ovejería



Insertar pocos elementos ordenadamente es barato

¿Cómo podemos usar esta noción para ordenar?

Insertion Sort

Para la secuencia inicial de datos, A :

1. Definir una secuencia ordenada, B , inicialmente vacía
2. Tomar el primer dato x de A
3. Sacar x de A e insertarlo en B donde le corresponda
4. Si quedan elementos en A , volver a 2.

Insertion Sort

Insertion Sort es correcto

Podemos demostrarlo por inducción

Finitud

En cada paso se saca un elemento de A

Cuando no quedan elementos en A , el algoritmo termina

La inserción requiere como máximo recorrer todo B

Como A y B son finitos, el algoritmo termina en tiempo finito

Correctitud

PD: Al terminar el i -ésimo paso, B se encuentra ordenada

Por **inducción** sobre i

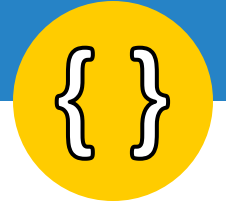
Caso Base: Después del paso 1, B tiene un solo dato $\rightarrow B$ está ordenada

Hipótesis Inductiva: Después del paso i , B está ordenada

Toca el paso $i + 1$. Extraemos el primer elemento de A , y lo insertamos ordenadamente en B . Termina el paso $i + 1$. Si la **inserción** fue correcta, entonces B está ordenada.

En particular, al terminar el algoritmo después del paso n , B está ordenada.

Inserción



Como se hace la **inserción** depende de la estructura

Se suele usar **arreglos**, pero también se puede usar **listas ligadas**

En cualquier caso, el algoritmo no necesita memoria adicional

Insertar



Primero, hay que buscar donde corresponde insertar el elemento

Luego hay que llevar a cabo la inserción

¿Cuál es la complejidad usando **arreglos**? ¿Y con **listas ligadas**?

Insertar en un Arreglo

El primer paso podemos hacerlo en $O(\log n)$ con búsqueda binaria

Para insertar hay que desplazar todos los elementos, lo que es $O(n)$

Por lo tanto, en **arreglos insertar** es $O(n)$

Insertar en una Lista Ligada

Para el primer paso es necesario revisar la lista entera, en $O(n)$

Teniendo el nodo donde corresponde insertar, hacerlo es $O(1)$

Por lo tanto, en **listas insertar** es $O(n)$

Complejidad de Insertion Sort

En cualquier caso, es necesario realizar n inserciones

Cada una cuesta $O(n)$

Por lo que la complejidad es $n \cdot O(n) = O(n^2)$

Actividad 1

Para la ayudantía de este viernes necesitan lo siguiente:

- Haber leído y seguido la guía de instalación de **C**
- ✓ Haber leído la guía de **arreglos**
- Haber leído la guía de **punteros**
- Traer su computador con todo listo para trabajar en C