

# Actividad 2.3

## *$T(n)$ de ordenación por Selección*

### Algoritmos y Estructuras de Datos

#### *Tema 2: algoritmos no recursivos*

Calcular rigurosamente la función de tiempo del algoritmo de “Ordenación por selección” proporcionado más abajo. Si lo necesitamos, usemos algunos resultados de la operación sumatorio (<http://es.wikipedia.org/wiki/Sumatorio>).

Aunque no es necesario saber lo que hace este algoritmo para poder analizarlo, puedes ver en qué consiste el algoritmo de ordenación por Selección en la animación que encontrarás en [https://es.wikipedia.org/wiki/Ordenamiento\\_por\\_selecci%C3%B3n](https://es.wikipedia.org/wiki/Ordenamiento_por_selecci%C3%B3n) (sólo fíjate en la animación, la cual es muy ilustrativa).

Los parámetros relevantes son:

- En el caso de “ordenarPorSeleccion()”, su parámetro relevante para calcular su función de tiempo es “n”, ya que es obvio que el algoritmo tarda más en ordenar un array cuanto mayor es éste (cuando mayor es “n”).
- En el caso de “buscarMinimo()”, vemos claramente que el algoritmo tardará más o menos dependiendo de la distancia entre inicio y fin, por lo tanto son estos dos parámetros los que debemos considerar para calcular la función de tiempo.

Haz el análisis indicando los números de línea que estás analizando, empezando obviamente desde dentro hacia fuera. Ejemplo:

1. De las líneas 23 a la 26 la función de tiempo es...
2. De las líneas 14 a la 31 la función de tiempo, por lo tanto, es...
3. La función de tiempo de la línea 5 por lo tanto es...
4. La función de tiempo de la 4 a la 11 por lo tanto es...
5. Por lo tanto, la función de tiempo de la 3 a la 12 es...
6. Por lo tanto, y como resultado final, la función de tiempo de la línea 2 a la 12 es  $T(n)=...$

```

1  // Ordena el vector de menor a mayor usando el método de selección
2  void ordenarPorSelección(int *vector, int n) {
3      for (int i=0; i<n; i++) {
4          // Buscamos el mínimo en lo que queda de vector hasta el final
5          int posicionMinimo = buscarMinimo(vector,n,i,n-1);
6
7          // Ponemos el mínimo en la posición adecuada del vector, y lo que
8          // había allí lo ponemos en donde encontramos el mínimo
9          int temp = vector[i];
10         vector[i] = vector[posicionMinimo];
11         vector[posicionMinimo] = temp;
12     }
13 }
14 // Busca el mínimo del rango [inicio,fin] de un vector y devuelve su posición
15 int buscarMinimo (int *vector, int n, int inicio, int fin) {
16     // Inicializamos el mínimo con el primer elemento
17     int minimo = vector[inicio];
18     int indiceMinimo = inicio;
19
20     // Cada vez que encontremos un número menor que el mínimo que tenemos,
21     // actualizamos el mínimo que tenemos con ese número
22     for (int i = inicio+1; i<=fin; i++) {
23         if (vector[i] < minimo) {
24             minimo = vector[i];
25             indiceMinimo = i;
26         }
27     }
28
29     // Devolvemos la posición del mínimo
30     return (indiceMinimo);
31 }

```

### Solución

- De 23 a 26 es  $T(\text{inicio}, \text{fin})=1$  porque todas las operaciones son elementales.
- De la 22 a la 27:
  - Las vueltas que da el bucle for son  $\text{fin}-(\text{inicio}+1)+1=\text{fin}-\text{inicio}$
  - Como cada iteración es  $T(\text{inicio}, \text{fin})=1$ , la función de tiempo total de la línea 22 a la 27 es  $T(\text{inicio}, \text{fin})=\text{fin}-\text{inicio}+1$  (el 1 del final son las operaciones elementales fruto de evaluar la última condición, que no se cumple, y el salto posterior hasta después del for)
- Las líneas 17, 18 y 30 son todas operaciones elementales. Por lo tanto, por la regla de la suma, `buscarMinimo` tiene  $T(\text{inicio}, \text{fin})=\text{fin}-\text{inicio}+1$
- De las líneas 4 a la 12:
  - Las líneas de 9 a 11 son todas operaciones elementales
  - La línea 5 es  $T(n)=n-1-i+1=n-i+1$ , pues:
    - `buscarMinimo()` dijimos que era  $T(\text{inicio}, \text{fin})=\text{fin}-\text{inicio}+1$
    - En la llamada a `buscarMinimo()`, `inicio` es `i` y `fin` es `n-1`
    - Además hay que sumar las operaciones elementales de la línea, las cuales se integran en el +1 del final
  - Por lo tanto, de 4 a 12 tenemos  $T(n)=n-i+1$
- De 3 a 12, usamos sumatorios (<http://es.wikipedia.org/wiki/Sumatorio>):

$$\begin{aligned}
1 + \sum_{i=0}^{n-1} (n+1-i) &= 1 + n^2 + n - \sum_{i=0}^{n-1} i = 1 + n^2 + n - \sum_{i=1}^{n-1} i \\
&= 1 + n^2 + n - \frac{(n-1)(n-1+1)}{2} = 1 + n^2 + n - \frac{1}{2}n^2 + \frac{1}{2}n \\
&= 0,5n^2 + 1,5n + 1
\end{aligned}$$

Fíjate en que, dentro del cálculo del sumatorio, no aplicamos las reglas de sustitución de constantes. Por eso eliminamos el -1 y el +1 que hay en el numerador de uno de los pasos.

- Finalmente, de la 2 a la 12, es el resultado anterior más algunas operaciones elementales (declaración de parámetros formales, salto del final de la función, etc.), las cuales se integran en el 1 que ya teníamos. Por lo tanto el resultado final es  $T(n)=0,5n^2+1,5n+1$