ALGORITMOS DE ORDENACIÓN

ALGORITMOS Y ESTRUCTURAS DE DATOS I

Algoritmos de ordenación

La ordenación es una aplicación fundamental en computación.

La mayoría de los datos producidos por un programa están ordenados de alguna manera y muchos de los cómputos son eficientes porque invocan internamente a un método de ordenación

Algoritmo de ordenación: Parte 1

- Realiza el ordenamiento o reordenamiento de una lista **a** de **n** valores
- La burbuja son dos términos de la lista consecutivos, que se comparan: si el primero es mayor que el segundo sus valores se intercambian.
- El número de recorridas que realiza en la lista solo depende de **n** (cantidad de elementos de la lista).

6 5 3 1 8 7 2 4

Algoritmo de ordenación: Burbuja (sin mejoras)

Implementación

- Consta de dos bucles anidados, uno con el índice **i**, que da un tamaño menor al recorrido de la burbuja y un segundo bucle con el índice **j** con un recorrido desde **0** hasta **n-i**, para cada iteración del primer bucle, que indica el lugar de la burbuja
- La burbuja son dos términos de la lista seguidos, **j** y **j+1**, que se comparan: si el primero es mayor que el segundo sus valores se intercambian.
- Esta comparación se repite en el centro de los dos bucles, dando lugar a una lista ordenada.

6 5 3 1 8 7 2 4

Algoritmo de ordenación: Burbuja (sin mejoras)

Eficiencia

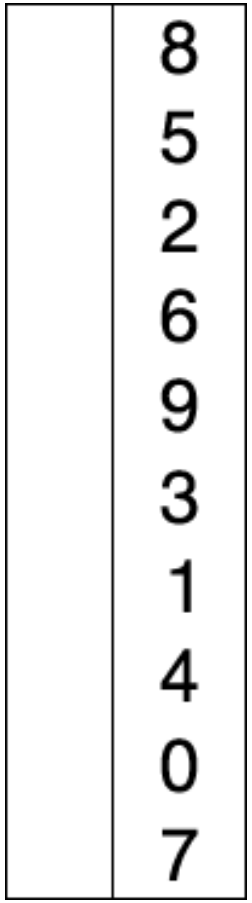
- Para ordenar un arreglo de n términos, tiene que realizar siempre el mismo número de comparaciones.

- El número de comparaciones **c(n)** no depende del orden de los términos, si no del número de términos.

- Mejora: se podría mejorar la eficiencia en un array parcialmente ordenado.

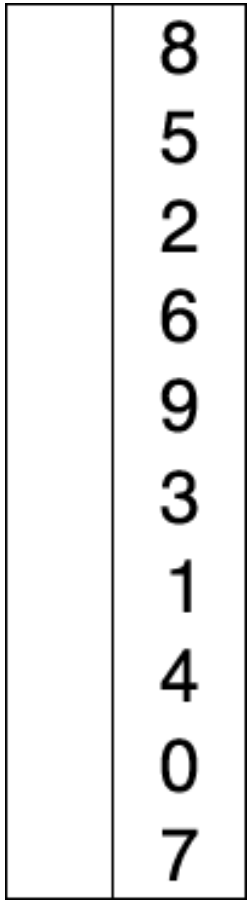
6 5 3 1 8 7 2 4

Algoritmo de ordenación: Parte 2



Algoritmo de ordenación: Parte 2

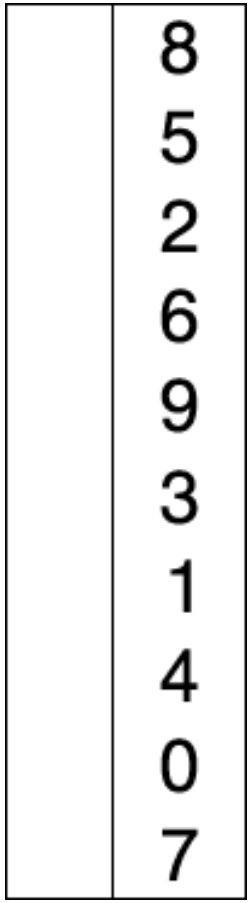
- Buscar el mínimo elemento de la lista
- Intercambiarlo con el primero
- Buscar el siguiente mínimo en el resto de la lista
- Intercambiarlo con el segundo
- Repetir procedimiento



Algoritmo de ordenación: Selection sort

Implementación

- Buscar el mínimo elemento entre una posición i y el final de la lista
- Intercambiar el mínimo con el elemento de la posición i

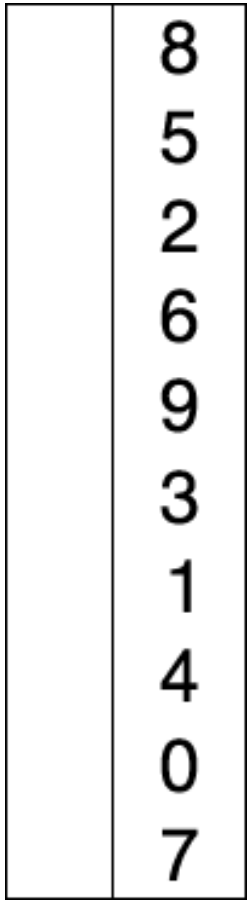


Algoritmo de ordenación: Selection sort

Eficiencia

- Este algoritmo mejora ligeramente el algoritmo de la burbuja. Esta mejora no es significativa si queremos ordenar una lista de enteros, pero si en caso de ordenar un vector de estructuras más complejas (costo de realizar más intercambios).

- No mejora su rendimiento cuando los datos ya están ordenados o parcialmente ordenados. El número de comparaciones c(n) no depende del orden de los términos, sino del número de términos



Algoritmo de ordenación: Parte 3

- Inicialmente contamos con un solo elemento, que es considerado un conjunto ordenado
- Cuando hay k elementos ordenados de menor a mayor, se toma el elemento k+1 y se compara con todos los elementos ya ordenados, deteniéndose cuando se encuentra un elemento menor.
- En este punto se inserta el elemento k+1 debiendo desplazarse los demás elementos.

Algoritmo de ordenación: Insert sort

- Consta de dos bucles anidados, uno con el índice **i**, que define el elemento que será comparada con el conjunto de elementos ya ordenados y un segundo bucle con el índice **j** con un recorrido desde **i-1 hasta 0.**
- Se irán corriendo los números siempre y cuando hacia la derecha, siempre y cuando el elemento de índice i sea menor al del conjunto ordenado.
- Al finalizar de recorrer el vector o al encontrara un número del conjunto ordenado que sea menor, se inserta el número de la posición i en el lugar siguiente al menor encontrado.

6 5 3 1 8 7 2 4

Algoritmo de ordenación: Insertion sort

Eficiencia

- Mejora su rendimiento cuando los datos están parcialmente ordenados.
- En el mejor de los casos, cuando el vector ya esta ordenado, el algoritmo solo hace una pasada.