UNIVERSIDAD AUSTRAL

Facultad de Ingeniería

ALGORITMOS Y ESTRUCTURAS DE DATOS

Trabajo práctico: Comparación de Algoritmos

Objetivos:

- Analizar algortimos.
- Usar la Interface Comparable.

Ejercicio 1

Modificar los algoritmos de búsqueda secuencial y búsqueda binaria de manera que devuelva la posición en donde se encuentra el valor a buscar. Tener en cuenta que el elemento a buscar puede no pertenecer al conjunto.

Ejercicio 2

Un problema muy frecuente es ordenar un conjunto de datos en forma ascendente o descendente. Los datos pueden ser números enteros, dobles, strings o estructuras más complejas. Este problema se conoce en inglés como sort. Algunos algoritmos básicos son selección, inserción, y burbujeo. Hay otros más complejos pero más eficientes como por ejemplo quicksort.

a) Describir los algoritmos de selección, inserción, burbujeo.
b) Codificarlos.
c) Generar en forma random un arreglo de números enteros para distintos valores de N (siendo N el tamaño del arreglo) y comparar los tiempos para distintos valores de N.
d) Modificar los algoritmos anteriores para que ordenen cadenas. Observar que hay que realizar muy pocos cambios.
e) Modificar los algoritmos de manera que ordenen cualquier tipo de dato (no sólo simples sino también compuestos, como por ejemplo debemos ordenar aviones por compañía y nro de avión).
f) Buscar temas afines en Internet (hay páginas muy interesantes).
g) Desarrollar la versión recursiva del método de selección.

Ejercicio 3

Otro problema frecuente es intercalar dos secuencias ordenadas. Este problema se conoce en inglés como merge. El enunciado es:
Dados dos arreglos ordenados, a y b, construir un tercer arreglo de salida c que contenga a los elementos de ambos arreglos de entrada, y esté ordenado.

Algoritmo de merge

- Mantener un índice para recorrer el arreglo a otro para b y otro para c.
- Recorrer linealmente los arreglos a y b, asignando de a un elemento por vez en el arreglo de salida c. El elemento a asignar en c es el menor entre el elemento actual de a y el actual de b.
- Incrementar en 1 el índice del arreglo del que provino el elemento y también incrementar en 1 el índice de c.

- Repetir hasta que uno de los dos arreglos de entrada haya sido pasado totalmente a c.
- Pasar el resto del otro arreglo a c.

Se pide codificar el algoritmo merge para cualquier tipo de objetos comparables.