Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática
Algoritmos y Estructuras de Datos Avanzadas

Curso 2023-2024

Práctica 5: Algoritmos de ordenación

---

1. Objetivo

El objetivo de esta práctica es trabajar los algoritmos de ordenación interna vistos en la teoría [1] y comprobar de forma empírica la complejidad computacional de las técnicas estudiadas. La implementación en lenguaje C++ utiliza la definición de funciones genéricas (plantillas), el polimorfismo dinámico y la sobrecarga de operadores.

2. Entrega

Se realizará en dos sesiones de laboratorio en las siguientes fechas:

Sesión tutorada: del 1 al 5 de abril
Sesión de entrega: del 8 al 12 de abril

3. Enunciado

Los métodos de ordenación son algoritmos que colocan los elementos de una secuencia dada según una relación de orden [2]. El resultado será un reordenamiento de la entrada que cumpla con esa relación de orden.

- Por simplicidad, asumimos que el registro de información almacenado en la secuencia coincide con el tipo de la clave utilizada para establecer su orden `Key`.
- Se pide implementar, al menos, los siguientes algoritmos de ordenación vistos en clase [1][2]:
  - Selección
  - QuickSort
  - HeapSort
  - Por Incrementos Decrecientes (ShellSort): debe permitir seleccionar la constante de reducción alfa, siendo 0 < alfa < 1
  - RadixSort

4. Notas de implementación

La implementación de los métodos de ordenación se realiza teniendo en cuenta las siguientes consideraciones:

1. La secuencia de valores a ordenar es una secuencia de tamaño fijo. Para almacenarla se utiliza la clase `staticSequence<Key>`, derivada a partir de la clase base `Sequence<Key>`, definida en la práctica 4 [3].
   a. En la clase base abstracta, `Sequence<Key>`, se añade la sobrecarga del operador de acceso:

```
virtual Key Sequence<Key>::operator[](const Position&) const = 0;
```

   b. La clase derivada, `staticSequence<Key>`, implementa el operador para permitir que los métodos de ordenación accedan a los elementos de la secuencia.

   c. Como se indica en la práctica 4 [3], el número máximo de valores que pueden

almacenarse en la secuencia, `Size`, se pasa como parámetro al constructor.

2. Se implementa una librería de métodos de ordenación:
   a. La implementación de cada método de ordenación se realiza mediante una **plantilla de función**, en la que se especifica el tipo de elementos a ordenar (`Key`). La función recibe como parámetros la secuencia de elementos de tipo `Key` a ordenar de elementos y su tamaño.

```
nombre_método<Key> (staticSequence<Key>, size)
```

3. Para utilizar el polimorfismo dinámico en el programa principal, se define también la siguiente familia de clases genéricas.

   a. La clase base abstracta `SortMethod<key>`, contiene como atributo un objeto `staticSequence<Key>` con la secuencia a ordenar.

   b. En esta clase se define un método nulo `Sort()`:

```
virtual void SortMethod<Key>::Sort() const = 0;
```

   c. Cada clase derivada implementa un método de ordenación: sobrecarga el método `Sort()` realizando una llamada a la plantilla de función adecuada.

4. El programa principal acepta las siguientes opciones por línea de comandos:

   a. `-size <s>`, `s` es el tamaño de la secuencia.

   b. `-ord <m>`, `m` es el código que identifica un método de ordenación.

   c. `-init <i> [f]`, indica la forma de introducir los datos de la secuencia

```
i=manual
i=random
i=file  f=nombre del fichero de entrada
```

   d. `-trace <y|n>`, indica si se muestra o no la traza durante la ejecución.

5. Se crea el objeto que implementa el método de ordenación, invocando al constructor con los parámetros indicados por línea de comandos. Se utiliza la clase `nif` definida en la práctica 4 [3] como tipo de dato `Key` en la plantilla .

6. El tipo de dato `nif` se actualiza con la sobrecarga de los operadores necesarios para el correcto funcionamiento de los métodos de ordenación.

7. Se rellena la secuencia según el parámetro indicado por línea de comandos.

8. A continuación se ejecuta el algoritmo de ordenación y se muestra el resultado de la operación. Si en el parámetro por línea de comando se elige mostrar la traza, se debe imprimir al menos la secuencia después de cada iteración del algoritmo.

5. Referencias

[1] Google: Apuntes de clase
[2] Wikipedia: Algoritmo de ordenamiento: https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento
[3] Aula virtual AyEDA: Práctica 4