Universidad de La Laguna

Diseño y Análisis de Algoritmos
Práctica 4 - Framework para algoritmos de Divide y Vencerás

Factor de ponderación: 25%

El patrón “Template”

El patrón “Template” o “Plantilla” permite la creación de clases genéricas que implementan diferentes algoritmos que deben ser completados por otras clases específicas. Así pues, la clase genérica crea un “esqueleto” con las partes básicas a realizar, teniendo en cuenta que algunas de las partes de ese esqueleto deben ser completadas **obligatoriamente** por la clase (o clases) específicas.

Esto proporciona un mecanismo flexible para el programador para implementar una serie de algoritmos del mismo tipo. Para la creación de un nuevo algoritmo específico, sólo será necesaria la implementación de una serie de métodos predefinidos, ya que el resto del proceso viene implementado en este “esqueleto”.

Para la utilización del patrón “Template” se hace uso de los mecanismos de herencia y polimorfismo de los lenguajes de Programación Orientada a Objetos, como son las clases y métodos abstractos o las interfaces. En los siguientes enlaces se muestran una serie de referencias que explican el patrón “Template”.

[1] https://es.wikipedia.org/wiki/Patr%C3%B3n_de_m%C3%A9todo_de_la_plantilla
[2] https://youtu.be/CAwxVo24L94?si=tZcJu9AWIFDECb0F

Los algoritmos de Divide y Vencerás

Tal y como se ha explicado en clase, los algoritmos de tipo Divide y Vencerás poseen una estructura común basada en 4 grandes operaciones:

- División del problema en subproblemas.
- Llamadas recursivas al algoritmo.
- Unión de las soluciones de los subproblemas para formar soluciones a problemas más grandes.
- Solución del problema cuando la instancia es lo suficientemente pequeña.

Así pues, todo algoritmo de Divide y Vencerás que divida su problema en 2 subproblemas (binario) tendría el siguiente pseudocódigo:

```
1 Solve(Problema p, int tamaño) {
2    if (Small(p))
3       return SolveSmall(p)
4    else {
5       Problema m[] = Divide(p, tamaño)
6       Solucion S1 = Solve(m[0], tamaño/2)
7       Solucion S2 = Solve(m[1], tamaño/2)
8       Solucion S = Combine(S1, S2)
9       return S
10   }
11 }
```

Si se quisiera implementar una clase que utilice el patrón “Template” para representar todos los algoritmos de Divide y Vencerás binarios, los elementos marcados en verde serían aquellas partes que deben ser delegadas al algoritmo específico.

Objetivos de la práctica

Los siguientes objetivos se consideran **condición necesaria pero no suficiente** para aprobar la práctica:

1. Crear un Framework que utilice el patrón “Template” para la implementación de algoritmos de **Divide y Vencerás binarios**.
2. Utilizar el paradigma de **Programación Orientada a Objetos**, así como los lenguajes de programación **C++, C#** o **Java**.
3. Implementar haciendo uso del Framework creado los algoritmos de **MergeSort** y **QuickSort**.

Requisitos evaluables de la práctica

Los siguientes requisitos se evaluarán de cara a la entrega de la práctica:

1. Todo el código deberá estar adecuadamente comentado y desarrollado atendiendo al paradigma de Programación Orientada a Objetos.

2. El Framework tiene que estar bien implementado de acuerdo al patrón “Template”.

3. Los algoritmos **MergeSort** y **QuickSort** deben hacer un buen uso de la plantilla. **NO** deben poder ejecutarse sin hacer uso de la plantilla.

4. Debe crearse un generador de instancias que permita la creación de vectores de enteros desordenados de diferentes tamaños.

5. Debe crearse un fichero principal (main) que realice los siguientes pasos:
    a. Genere un conjunto de instancias aleatorias de diferentes tamaños.
    b. Para cada instancia, ejecute los algoritmos **MergeSort** y **QuickSort**.
    c. Muestre por pantalla (y opcionalmente por fichero) una tabla con la comparación de tiempos de ejecución de los algoritmos para diferentes tamaños de instancia.

6. El Framework debe poseer mínimo 2 métodos:
    a. Un método que permita la ejecución del algoritmo y devuelva su solución.
    b. Un método que devuelva una cadena de caracteres que represente la recurrencia del algoritmo en cuestión. Ésta debe ser de la forma **T(n) <= aT(b) + c**, donde **a, b** y **c** son devueltos por el algoritmo específico.

7. Tanto las soluciones como las instancias deben estar preparadas para la utilización de otros algoritmos además de los solicitados.

Opcional:
Como trabajo opcional de cara a mejorar la nota, se pide implementar el Framework para que sea compatible con algoritmos Divide y Vencerás **no binarios** (es decir, que puedan partirse en más de 2 subproblemas).

Durante la defensa de la práctica **se podrá solicitar algún tipo de modificación o prueba adicional**, la cual afectará en diferente grado a la nota final.

Entrega de la práctica

La práctica debe entregarse en tiempo y forma acorde a lo indicado en la tarea del campus virtual. Para poder considerar la práctica como aprobada, deben cumplirse 2 requisitos:

- La práctica debe defenderse en su sesión correspondiente. Además, debe funcionar tal y como se especifica en el presente enunciado.
- Se debe entregar la tarea del campus virtual incluyendo un fichero con extensión **tar.gz** con el código fuente del programa.

La no realización de uno de estos 2 puntos conlleva la calificación como suspenso de la práctica.