**Trabajo Práctico Obligatorio**

**Programación III**

Profesores:

<…>

Grupo:

<…>

<Fecha>

**Tabla de Contenidos**

# Enunciado

Para el trabajo práctico se deberá resolver en lenguaje Java un ejemplo de cada una de las técnicas vistas en clase en conjunto con un análisis teórico de la complejidad y una verificación empírica de la misma.

1. División y conquista

*Dados n puntos de la forma (x,y), se pide encontrar el par de puntos con la distancia más pequeña entre ellos.*

* *Crear el paquete ejercicio1*
* *Realizar una clase Punto que tenga las coordenadas (x,y) un constructor*
* *El algoritmo debe devolver una clase SoluciónEjercicio1 que tiene dos puntos y un entero con la cantidad de instrucciones ejecutadas*
* *Un método estático Ejercicio1 que reciba una lista de puntos y devuelva un objeto de la clase SoluciónEjercicio1*
* *Una prueba unitaria que a partir de 32 puntos pre-generados verifique que devuelva los puntos esperados y verifique que la cantidad de instrucciones ejecutadas*

2. Greedy

Resolver el problema de selección de actividades. Es un problema de optimización combinatoria donde se deben seleccionar, dentro de un marco de tiempo determinado, actividades de modo que no exista superposición entre las mismas. De cada una se conoce un instante de inicio (si) y un instante de finalización (fi). El problema es seleccionar el número máximo de actividades que pueden ser realizadas asumiendo que sólo puede trabajar en una sola actividad a la vez.

* *Crear el paquete ejercicio1*
* *Realizar una clase Tarea que tenga un entero de Inicio y otro de Fin*
* *El algoritmo debe devolver una clase SoluciónEjercicio2 que tiene la cantidad de tareas a ejecutar puntos y un entero con la cantidad de instrucciones ejecutadas*
* *Un método estático Ejercicio 2 que reciba una lista de Tareas y devuelva un objeto de la clase SoluciónEjercicio2*
* *Una prueba unitaria que a partir de 16 tareas pre-generadas verifique que devuelva los puntos esperados y verifique que la cantidad de instrucciones ejecutadas*

3. Greedy con Grafos

Resuelva el algoritmo de Kruskal visto en clase.

* *Crear el paquete ejercicio3*
* *Realizar una clase Vértice que tenga un nombre*
* *Realizar una clase Arista que tenga el vértice origen, el destino y el peso*
* *Realizar una clase Grafo que tenga una lista de vértices y de aristas*
* *El algoritmo debe devolver una clase SoluciónEjercicio3 que tiene el valor de la sumatoria de las aristas del spanning tree y un entero con la cantidad de instrucciones ejecutadas*
* *Un método estático Ejercicio 3 que reciba una Grafo y devuelva un objeto de la clase SoluciónEjercicio3*
* *Una prueba unitaria que a partir de un grafo pre-generado devuelva el valor esperado y verifique que la cantidad de instrucciones ejecutadas*
* *Resolver en un método aparte lo del unir particiones!*

4. Programación dinámica. Resolución por backtracking.

Resuelva el algoritmo de la mochila visto en clase en su versión backtracking.

* *Crear el paquete ejercicio4*
* *Realizar una clase ElementoMochila que tenga un peso y una ganancia*
* *El algoritmo debe devolver una clase SoluciónEjercicio4 que tiene el valor de la máxima ganancia a obtener y un entero con la cantidad de instrucciones ejecutadas*
* *Un método estático Ejercicio4 que reciba una lista de ElementoMochila y devuelva un objeto de la clase SoluciónEjercicio4*
* *Una prueba unitaria que a partir de una lista de ElementoMochila pre-generada devuelva el valor de la ganancia máxima esperada y verifique que la cantidad de instrucciones ejecutadas*

5. Programación dinámica. Resolución por programación dinámica.

Resuelva el algoritmo de la mochila visto en clase en su versión por programación dinámica.

* *Crear el paquete ejercicio5*
* *Realizar una clase ElementoMochilaDyn que tenga un peso y una ganancia*
* *El algoritmo debe devolver una clase SoluciónEjercicio4 que tiene el valor de la máxima ganancia a obtener y un entero con la cantidad de instrucciones ejecutadas*
* *Un método estático Ejercicio4 que reciba una lista de ElementoMochilaDyn y devuelva un objeto de la clase SoluciónEjercicio5*
* *Una prueba unitaria que a partir de una lista de ElementoMochilaDyn pre-generada devuelva el valor de la ganancia máxima esperada y verifique que la cantidad de instrucciones ejecutadas*

6. Backtracking

Dado una lista de números, encontrar N particiones no vacías de esos números que sumen lo mismo. Es posible que dejar números fueras de alguna de las particiones.

* *Crear el paquete ejercicio6*
* *El algoritmo debe devolver una clase SoluciónEjercicio6 que tiene el valor de la cantidad de soluciones correctas y un entero con la cantidad de instrucciones ejecutadas*
* *Un método estático Ejercicio6 que reciba una lista de enteros y devuelva un objeto de la clase SoluciónEjercicio6*
* *Una prueba unitaria que a partir de una lista de 8 números pre-generada verifique la cantidad de soluciones correctas y verifique que la cantidad de instrucciones ejecutadas*

7. Entrebables

De cada Ejercicio se pide entregar un documento digital con

* Pseudocódigo
* Código en Java
  + El código de java debe tener una variable que cuenta instrucciones para realizar una comprobación empírica de la complejidad teórica
    - No contar CADA instrucción, sino incluir un contador significativo para verificar la complejidad teórica
* Análisis teórico de la complejidad
* Prueba Unitaria donde:
  + Se verifica su funcionamiento a partir de un set de datos de entrada
  + Se verifica que la cantidad de instrucciones ejecutadas coincide con la esperada

Además se deben entregar los .java con fuentes de los ejercicios y las pruebas unitarias.

8. Entregas

Las entregas se realizan semanalmente a partir de la entrega del enunciado. Se recomienda avanzar con todos los ejercicios cuanto antes. El cronograma de entregas se corresponde a la forma en que se realizarán las correcciones

* Primer Entrega (9 Mayo)
  + Estructura del proyecto eclipse y primer ejercicio resuelto
* Segunda Entrega (23 Mayo)
  + Segundo y tercer ejercicio resuelto (06 Junio)
* Tercer Entrega (13 Junio)
  + Cuarto y quito ejercicio resuelto
* Curta entrega (27 Junio)
  + Sexto Ejercicio Resuelto