Departamento de
Informática e Ingeniería
de Sistemas

Universidad Zaragoza

Prácticas de Algoritmia para problemas difíciles
Especialidad en Computación, grado en Ingeniería Informática

Curso 2014-2015

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas
Area de Lenguajes y Sistemas Informáticos

14 de octubre de 2014

Organización general de las prácticas.

Se formarán equipos de una o dos personas. Si una de las dos abandona la asignatura, la otra deberá terminar en solitario.

Las prácticas se realizarán en el computador `hendrix`.

El lenguaje para la implementación puede ser Java o cualquier otro elegido por el alumno (incluyendo en la documentación todos los detalles necesarios sobre el mismo: versión, compilador, etc).

La entrega de la práctica X (X = 1, 2, ...) se realizará en `hendrix` mediante la ejecución de `someter apd_14 practicaX.tar`, siendo la fecha límite una semana antes de la fecha del examen correspondiente a la convocatoria que se presenten los miembros del grupo. Además, habrá que concertar una cita con la profesora para explicar las prácticas: descripción general del programa, elaboración, funcionamiento, demostración, etc.

El fichero `practicaX.tar` contendrá **un directorio denominado practicaX** con los ficheros de texto incluyendo:

- Descripción general del programa: cómo está organizado, qué se puede y qué no se puede hacer (tiene que llamarse **LEEME**). Contendrá en sus primeras líneas la lista de integrantes del grupo, con el siguiente formato:

  ```
  Apellido1 Apellido2, Nombre [tab] correo@electronico [tab] login en
  hendrix

  Apellido1 Apellido2, Nombre [tab] correo@electronico [tab] login en
  hendrix
  ```

  Donde [tab] representa el carácter tabulador.

- Listados del código debidamente comentados y dispuestos para ser compilados y utilizados.
- Un programa para el shell `ejecutarX.sh` que automatice la compilación y ejecución de algunos casos de prueba para los programas entregados. Deberá funcionar en hendrix.
- Los ficheros auxiliares de entrada necesarios para ejecutar las pruebas del punto anterior.

En la calificación se tendrán en cuenta los siguientes aspectos: documentación, funcionamiento e implementación.

El diseño ha de ser modular, basado en el uso de tipos abstractos de datos, con todas las funciones correctamente especificadas.

Las reglas generales de tratamiento de casos de plagio de la asignatura se aplicarán, en particular, a todas las prácticas.

Práctica 1

Vamos a desarrollar un "VC solver" o resolvedor de VC (Covertura de Vértices) que dado un grafo G encuentre un cubrimiento U lo menor posible de G.

Se trata del problema visto en clase, y es importante recordar que se trata de un problema NP-completo y que la resolución de VC es un reto en el que trabajan miles de informáticos de primer nivel. Por tanto hay que tener en cuenta lo siguiente

- Cualquier fragmento de código o idea para el que se utilicen fuentes externas debe ser identificado y dichas fuentes citadas.
- En la práctica hay que hacer como mínimo lo que se especifica a continuación, pero la tarea completa no está limitada, por lo tanto los más ambiciosos pueden necesitar fijarse un límite personal.

1.1. ¿Qué hay que hacer?

Hacer un programa que resuelva VC (encontrando un cubrimiento que puede ser mínimo o no) en todos los casos, separando al menos los siguientes:

1. Los grafos bipartitos (usados en el tema de Programación lineal y reducciones de Algoritmia Básica, la definición está en cualquier libro de grafos).
2. Los árboles, es decir, grafos acíclicos.

En los casos 1. y 2. anteriores el programa debe encontrar la solución óptima y funcionar eficientemente (en tiempo polinómico).

Debe utilizarse al menos una heurística no trivial (por ejemplo, algoritmos genéticos o "simulated annealing" (recocido simulado)).

El programa se llamará `VCsolver` y tendrá al menos las siguientes opciones:

1. Leer el grafo como matriz de adyacencia o como lista de aristas.
2. Introducir el grafo por medio de un menú explicativo.

3. Introducir un parámetro k cota superior del tamaño del cubrimiento.
4. Decir si se trata de un grafo bipartito o un árbol.
5. Devolver un cubrimiento y su tamaño.
6. Decir si existe o no un cubrimiento de tamaño menor o igual que k.

1.2. Entrega

Deberá entregarse una semana antes de la fecha del examen.