Departamento de
Informática e Ingeniería
de Sistemas

Universidad Zaragoza

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

Curso 2022-2023

Elvira Mayordomo y Ubaldo Ramón
Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas
Area de Lenguajes y Sistemas Informáticos

21 de septiembre de 2022

Organización general de las prácticas.

Se formarán equipos de una o dos personas, según se declare junto con la entrega de la primera práctica. Si una de las dos personas abandona la asignatura, la otra deberá terminar en solitario.

Las prácticas se realizarán en el computador `hendrix` y el lenguaje para la implementación será preferentemente `C++`. Excepcionalmente pueden aceptarse otras alternativas siempre que puedan ejecutarse directamente en Cygwin o en Windows 10 (incluyendo en la documentación todos los detalles necesarios: versión, compilador, etc).

La entrega de la práctica X (X = 1, 2, ...) se realizará en moodle mediante un fichero comprimido, siendo la fecha límite la indicada al final de cada enunciado. Si es necesario la profesora citará a los componentes de alguno o todos los equipos para explicar alguna o todas las entregas realizadas.

El fichero entregado contendrá **un directorio denominado practicaX** con los ficheros necesarios incluyendo:

- Una memoria en formato PDF en la que se incluya un informe completo con al menos:
  - Una descripción de qué se ha implementado y usando qué fuentes (web, libros, artículos, código propio, etc). Cualquier decisión tomada sobre implementación y sus razones. Si hay algo que decir sobre el lenguaje de programación utilizado.
  - Una explicación completa sobre la complejidad en tiempo de la implementación realizada y las prestaciones de la misma. El tiempo se puede medir a priori y/o experimentalmente y puede ser caso peor, tiempo probabilista u otras opciones mientras estas se justifiquen. Las prestaciones deben incluir cualquier error probabilista o de aproximación, heurística utilizada u otros, en caso necesario indicando e incluyendo los casos de prueba aceptados en el estado del arte.
  - Cómo se ha repartido el trabajo entre los dos integrantes del equipo.
  - Qué pruebas se han hecho, de dónde se han sacado los datos, resultados obtenidos (incluyendo eficiencia).
- Descripción general del programa en fichero de texto: 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 (excepcionalmente en Cygwin/Windows 10) .
- Los ficheros auxiliares de entrada necesarios para ejecutar las pruebas del punto anterior.
- Los ficheros de otras pruebas realizadas.

En la calificación se tendrán en cuenta los siguientes aspectos: completitud de la memoria, 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: ingeniería algorítmica y ordenación

La ingeniería algorítmica pretende transferir la innovación algorítmica a las aplicaciones y en particular trabaja en la implementación y evaluación experimental de cara a desarrollo de algoritmos eficientes. Podéis leer el artículo introductorio [1] y en particular sus secciones "Experimentos" e "Instancias y benchmarks". (Nota: Para descargar el artículo hay que estar en el dominio de unizar).

Algunas ideas importantes que aparecen en el artículo son que la experimentación algorítmica está enfocada a las hipótesis refutables, es decir, el que una serie de instancias funcionen de una cierta manera no implica que lo hagan otras, pero si una serie de instancias no se comporta de una forma A eso descarta que el algoritmo se comporte de forma A en todos los casos.

Otra de las ideas que nos interesan es que la forma de experimentar con distintos algoritmos, es decir, la poda que se hace del espacio de todos los experimentos posibles, puede depender mucho del algoritmo en sí. Por ejemplo un algoritmo que se comporte exactamente igual para ordenar cualquier vector de un tamaño dado no necesita demasiadas pruebas. Dos de las posibilidades más obvias, entradas generadas aleatoriamente y entradas reales o realistas, pueden ser más o menos adecuadas en función del algoritmo en sí y en cualquier caso dan conclusiones muy distintas sobre la distancia entre el análisis teórico a priori y el uso en la práctica del mismo algoritmo. Por ejemplo, el artículo dice "Las instancias aleatorias generadas ingenuamente tienden a ser mucho más fáciles o difíciles que las entradas realistas"[1].

1.1. ¿Qué hay que hacer?

1. Implementar los algoritmos de ordenación radix sort, quick sort y un tercero de tu elección.

   Para concretar el uso de radix sort todos los datos a ordenar (o las claves a ordenar de dichos datos) serán números enteros (que en función del tamaño de la entrada que se quiera probar pueden no ser representables/almacenables como enteros predefinidos en C++).

   No es necesario que las implementaciones sean propias pero sí que debe identificarse cualquier fragmento de código o idea para el que se utilicen fuentes externas, citando dichas fuentes.

[1] ¿Estás de acuerdo con esta afirmación?

Es importante que las implementaciones de los tres algoritmos sean homogéneas, de forma que las prestaciones de los mismos sean comparables. Si se utilizan fuentes externas para el código este aspecto debe tenerse especialmente en cuenta.

2. Elaborar y/o recopilar cuantos conjuntos de datos de prueba se consideren necesarios y con los tamaños que se pretendan resolver. Este apartado se valorará especialmente.

3. Probar y comparar las prestaciones de los tres algoritmos sobre cada conjunto de prueba. Escribir un informe con las conclusiones alcanzadas.

1.2. Entrega

Deberá entregarse **hasta el día anterior a la tercera sesión de prácticas**.

Bibliografía

[1] Peter Sanders Algorithm Engineering – An Attempt at a Definition Using Sorting as an Example *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, pp.55–61, 2010. https://doi.org/10.1137/1.9781611972900.6