Métodos algorítmicos en resolución de problemas

Grado en Ingeniería Informática (UCM). **Profesor:** Ricardo Peña

Prácticas opcionales del primer cuatrimestre — Curso 2019/2020

---

1. Prácticas propuestas

1. Implementar o Java o en C++ un árbol de búsqueda autoajustable con las operaciones de buscar, insertar y borrar una clave.

2. Implementar o Java o en C++ un árbol 2-3-4 con las operaciones de buscar, insertar y borrar una clave.

3. Implementar o Java o en C++ un montículo binomial con el la unión, el mínimo, la inserción, el borrado del mínimo, y decrecer una clave suponiendo que se tiene un puntero al nodo de dicha clave.

4. Implementar o Java o en C++ un algoritmo, que dado un grafo dirigido, detecte si tiene o no ciclos. En caso de ser acíclico, ha de listar sus vértices en orden topológico. Si hay más de uno posible, los puede listar en cualquiera de ellos. En caso de ser cíclico, ha de listar sus componentes fuertemente conexas como conjuntos de vértices.

5. Implementar o Java o en C++ la versión del algoritmo de Dijkstra de coste en $\Theta((a+n)\log n)$ que usa un montículo. Programar para ello un montículo sesgado con la operación adicional de *decrecer-clave*.

6. Implementar o Java o en C++ el algoritmo de Kruskal junto con una estructura de partición con compresión de caminos.

7. Implementar o Java o en C++ la versión del algoritmo de Prim de coste en $\Theta((a+n)\log n)$ junto con un montículo zurdo que soporte la operación de *decrecer-clave*.

8. Implementar o Java o en C++ el algoritmo de Bellman-Ford de caminos mínimos (ver libro de Cormen, Cap. 24). Deben ejecutarse ejemplos con y sin ciclos de coste negativo.

2. Normas para la presentación de la práctica

1. Sólo se podrá escoger una de las propuestas.

2. Debe comunicarse al profesor la intención de hacerla **a lo sumo el 22 de noviembre** mediante un mensaje dentro del Campus Virtual. El profesor las asignará por orden de petición y las confirmará sobre la marcha.

3. Serán individuales.

4. La entrega se hará **a o sumo el 8 de Enero**, a través del Campus Virtual, con un texto fuente que pueda ser compilado y ejecutado, ficheros de prueba, y una memoria de dos a cinco páginas que incluya algunos **casos de prueba** sencillos y las gráficas de tiempo de los casos de prueba voluminosos.

5. Es **obligatorio** presentar **gráficas con tiempos** para diferentes tamaños de la estructura, y deben discutirse los costes obtenidos con respecto a los previstos por la teoria.

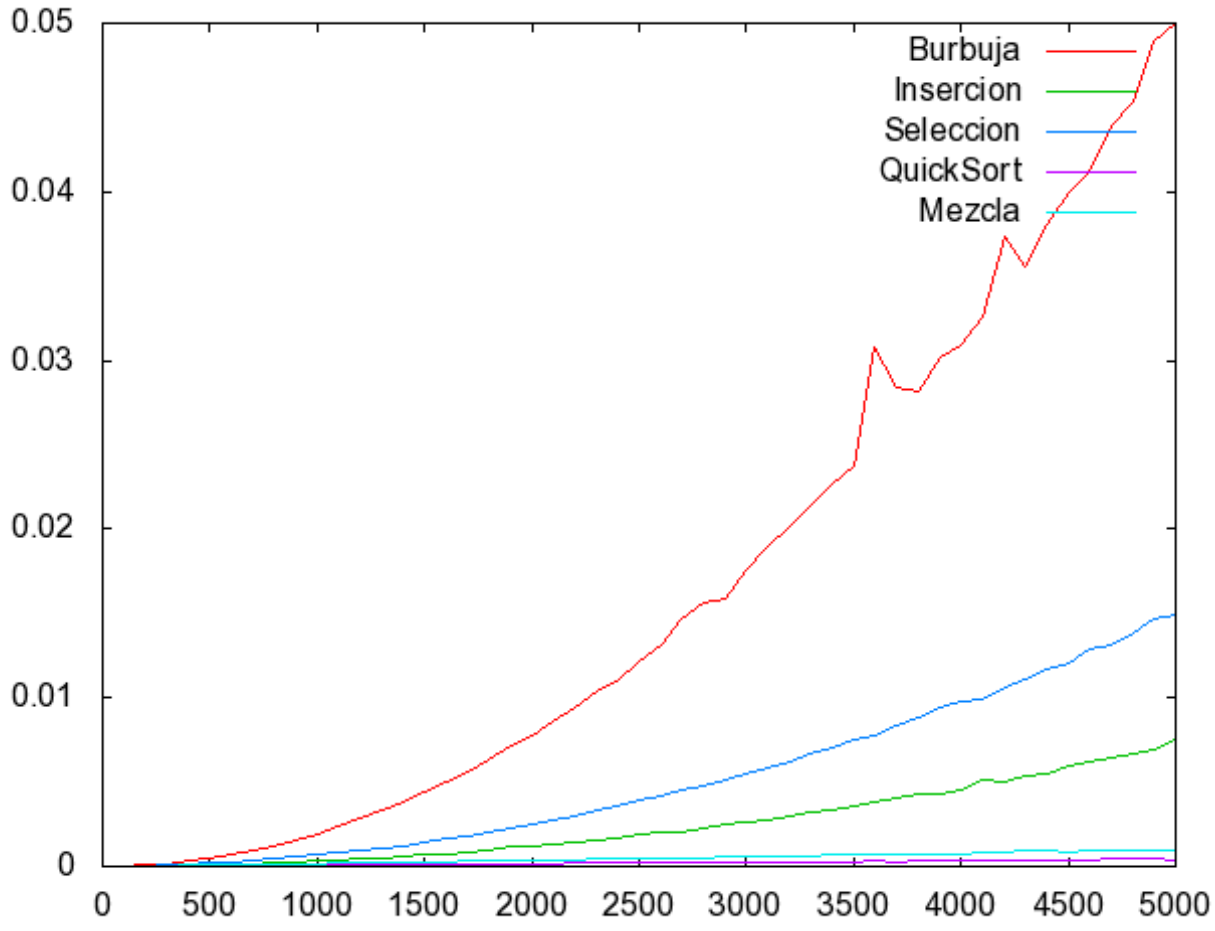
6. Se recomienda no emplear esfuerzo en interfaces gráficas o sofisticadas. Se valorará esencialmente la limpieza, claridad y eficiencia del código. También la **inclusión de suficientes comentarios**.

7. El alumno/a podrá ser llamado a consulta para presentar la práctica y responder a las preguntas del profesor.

8. Cada práctica se valora de 0 a 10. Si recibe una nota inferior a 4, no afecta a la nota de curso. A partir de 4 o más, representará el 20 % de la nota de curso del cuatrimestre.

3. Creación de gráficas

Se recomienda usar la librería de libre distribución GnuPlot (`http://www.gnuplot.info/`), la cual recibe ficheros con tablas bidimensionales de datos numéricos y produce gráficas como la siguiente:



Para medir los tiempos con fiabilidad, se deben seguir las siguientes normas:

- No deben medirse tiempos por debajo 10 milisegundos. Si el caso medido tarda menos, hacer que se ejecute 100 o 1000 veces y dividir el tiempo obtenido por esa cantidad.
- Usar preferiblemente las facilidades de medición del sistema operativo, no las de la librería del lenguaje.
- Ejecutar el caso en condiciones de baja o nula carga de la máquina (desconectarla de Internet y asegurarse de que se ejecutan pocos procesos)
- Para compensar las fluctuaciones, hacer siempre tres medidas y calcular la media de las tres.