Universidad de Valladolid
Departamento de Informática

Escuela de Ingeniería Informática | Campus Miguel Delibes, s/n. 47011 Valladolid | Tel.: 983423670 Fax: 983423671

Análisis y Diseño de Algoritmos

Grado en Ingeniería Informática

Práctica 1 - curso 2022/23

Introducción

El algoritmo de ordenación Quicksort es uno de los más populares. Existen variantes del algoritmo de ordenación Quicksort que lo mejoran. El objetivo de la práctica es implementar una versión modificada del algoritmo Quicksort que incorpore dos modificaciones sobre el algoritmo original y comparar el algoritmo resultante con el algoritmo original. Las modificaciones a implementar son:

- Como con la mayoría de los algoritmos recursivos, el algoritmo Quicksort no obtiene buenos resultados con vectores pequeños. La variante consiste en que a partir de un determinado valor umbral K (cuando los vectores que hay que ordenar son menores de un cierto tamaño) se reemplaza la llamada recursiva a Quicksort por otro algoritmo que se comporte mejor con este tipo de vectores pequeños. El algoritmo a utilizar para vectores pequeños será al algoritmo de ordenación por inserción.

  Por tanto, el algoritmo Quicksort modificado deberá emplear ordenación por inserción para vectores con menos de K elementos. Se deberá determinar de manera empírica el mejor valor de K para el que el algoritmo realiza menos operaciones, según se indica en la parte 1 de la práctica.

- El funcionamiento del algoritmo Quicksort se basa en la elección del pivote, el cual sirve de referencia para hacer las particiones. Es por esto que la elección del pivote es uno de los aspectos más importantes de los que hay que ocuparse cuando se utiliza este método de ordenación, ya que el rendimiento puede variar de forma sustancial en función de que se haya elegido un buen o un mal pivote.

  En el algoritmo Quicksort original se emplea como pivote el primer elemento del vector a ordenar. La modificación que se propone deberá emplear como pivote la mediana de tres elementos: el que ocupa la primera posición, el que ocupa la posición central y el que ocupa la última posición.

Desarrollo de la práctica

La realización de la práctica constará de dos partes.

Parte 1

En la primera parte de la práctica se determinará de manera empírica el mejor valor para el umbral K, de manera que el algoritmo realice el menor número de operaciones. Para ello, en primer lugar, se implementará el algoritmo Quicksort modificado, que incluirá las dos modificaciones descritas en el apartado de introducción. A continuación se usarán datos experimentales para determinar el valor umbral K que indica cuando se empleará la ordenación por inserción. Para ello se empleará una estimación de la eficiencia **en el caso promedio** del algoritmo.

- Hay que construir un programa para obtener los datos que contará las operaciones elementales que ejecute el algoritmo.

- Se analizarán distintos valores del umbral *K*, entre 3 y 30. Se crearán y asignarán aleatoriamente vectores de un tamaño fijo (por ejemplo, 100.000). Para cada valor del umbral *K* se llamará un cierto número de veces (por ejemplo 20) al algoritmo de ordenación. Debe tenerse en cuenta que antes de cada llamada al subprograma se deberá volver a asignar valores aleatorios al vector. Estos vectores aleatorios se crearán del siguiente modo:

```
1. Rellenar cada vector V de tamaño M con los valores 1, 2, ..., M
2. Descolocar los elementos:
      para i=1 hasta M
        x=random();
        y=random();
        intercambiar(V[x],V[y])
      fin_para
```

- El programa servirá para obtener los datos necesarios para analizar el algoritmo. Proporcionará como salida las tablas que indiquen el número de comparaciones y el número de asignaciones promedio para cada valor del umbral *K*.

Con los datos anteriores se construirán dos gráficas, una para el número de comparaciones y otra para el número de asignaciones. A partir de ellas se intentará determinar **cuál es el valor óptimo del umbral *K*** para el algoritmo Quicksort modificado.

Parte 2

Se compararán el algoritmo Quicksort original y el algoritmo Quicksort modificado, empleando el valor del umbral *K* seleccionado en la primera parte de la práctica. Para ello es preciso obtener una estimación, basada en datos experimentales, de la eficiencia **en el caso promedio** de los algoritmos.

- Hay que construir un programa para obtener los datos que contará las operaciones elementales que ejecute el algoritmo y medirá el tiempo empleado.
- Se crearán y asignarán aleatoriamente vectores de distintos tamaños. Por ejemplo 10.000, 20.000, . . . , 100.000. Para cada tamaño se llamará un cierto número de veces (por ejemplo 20) al algoritmo de ordenación. Debe tenerse en cuenta que antes de cada llamada al subprograma se deberá volver a asignar valores aleatorios al vector. Se emplearán vectores **"casi ordenados"**. Estos vectores se crearán del siguiente modo:

```
1. Rellenar cada vector V de tamaño M con los valores 1, 2, ..., M
2. Descolocar los elementos:
      para i=1 hasta M/10
        x=random();
        y=random();
        intercambiar(V[x],V[y])
      fin_para
```

- El programa nos servirá para obtener los datos necesarios para analizar el algoritmo. Proporcionará como salida las tablas que indiquen el número de comparaciones promedio, el número de asignaciones promedio y el tiempo promedio empleado según el tamaño del vector.

Con los datos anteriores se construirán tres gráficas, una para el número de comparaciones, otra para el número de asignaciones y otra para el tiempo empleado. También hay que determinar, si fuera posible, la fórmula que más se aproxima a los datos obtenidos y que los describa en función del tamaño del vector. Por último, hay que **determinar cuál de los dos algoritmos es el más eficiente**.

Contar operaciones elementales

Para contar las operaciones elementales se añadirán las instrucciones necesarias en el programa para contar el número promedio de comparaciones y el número promedio de asignaciones. Se considerarán solamente las comparaciones y asignaciones **en las que intervengan elementos del vector** (no las comparaciones y asignaciones entre índices, por ejemplo).

Medir el tiempo empleado

La medición del tiempo se puede realizar de varias maneras. Una de las más sencillas es emplear las clases *Instant* y *Duration* de Java.

```
import java.time.Duration;
import java.time.Instant;
...
Instant start = Instant.now();
// Código que queremos medir
Instant finish = Instant.now();
long timeElapsed = Duration.between(start, finish).toNanos();
```

En ocasiones la medición del tiempo no es muy precisa, sobre todo cuando el tamaño del vector es pequeño. Por tanto, se recomienda utilizar tamaños de vector grandes.

Análisis de datos

Para analizar los datos se sugiere utilizar programas que permitan representar gráficamente los datos obtenidos, por ejemplo LibreOffice Calc, Microsoft Excel o cualquier herramienta conocida que sea útil para este tipo de tareas.

Normas de entrega

La práctica se realizará en **grupos de 2 personas**.

Se deberá entregar el código fuente de los programas que se han creado para la realización de la práctica (se recomienda el uso del lenguaje Java aunque alternativamente se permite realizar en cualquier lenguaje de programación). Dichos programas deben compilar y ejecutar perfectamente en las máquinas de los laboratorios. Estos programas deben generar como salida un fichero con los datos promedio.

Además se entregará un documento en **formato PDF** donde se indique lo siguiente:

- Nombre de los alumnos.
- **Muy importante**: hay que estimar como se ha repartido el trabajo entre los miembros del equipo. Para ello, **hay que indicar el porcentaje de esta entrega que se estima ha desarrollado cada miembro del equipo.**

- Una breve descripción de cómo se ha realizado la práctica (rango de tamaños y repetición de medidas, programas utilizados para el análisis, etc.).
- Las tablas y los gráficos con los resultados obtenidos en las medidas experimentales de los algoritmos.
- El valor umbral *K* que se utilizará en la versión del algoritmo Quicksort modificado.
- La fórmula obtenida para cada uno de los dos algoritmos.
- **Una explicación RAZONADA de qué algoritmo es mejor.**

La fecha límite de entrega es el **21 de octubre a las 14:00**. Los ficheros de la práctica se empaquetarán en un fichero zip cuyo nombre será `p1_grupoX.zip`, siendo X el número del grupo de laboratorio. El fichero zip se entregará a través del Campus Virtual (`https://campusvirtual.uva.es/`).

La defensa de la práctica es obligatoria y se realizará en la sesión de laboratorio de la semana siguiente a la entrega:

- Grupo X1: miércoles 26 de octubre a las 10:00.
- Grupo X2: jueves 27 de octubre a las 12:00.
- Grupo X3: viernes 28 de octubre a las 11:00.