CC3001 Algoritmos y Estructuras de Datos
Tarea 1: Algoritmos simples de ordenación

Profs. Nelson Baloian, Benjamin Bustos, José A. Pino
4 de agosto de 2009

Fecha de entrega: 18 de agosto de 2009 a las 23:59:59

Objetivos

El objetivo de esta tarea es realizar una pequeña investigación experimental, basada en tres algoritmos de ordenación utilizando algoritmos iterativos. Para realizar la implementación de los algoritmos se debe utilizar el lenguaje Java (disponible en las estaciones de trabajo del DCC). No se permite el empleo de otro lenguaje de programación.

Descripción de la tarea

Deben analizar e implementar tres métodos de ordenación sencillos:

i. Burbuja.

ii. Shellsort.

iii. Combsort.

Algoritmo Burbuja

Visto en cátedra.

Algoritmo Shellsort

El algoritmo de ordenamiento Shellsort funciona de manera similar al algoritmo de inserción (visto en cátedra), pero entre elementos separados a una distancia h que va disminuyendo en cada iteración. Lo que se obtiene

con esto es que los elementos del arreglo ya están relativamente ordenados cuando se realiza la última iteración con $h = 1$, que es equivalente al algoritmo de inserción. Para esta tarea, utilice la siguiente serie de pasos:

$$h_i = \frac{3^i - 1}{2} \ (h = \{1, 4, 13, 40, 121, \ldots\})$$

en donde el paso mayor es el número de la serie que se encuentra dos posiciones antes del más cercano a n.

Algoritmo Combsort

Este algoritmo utiliza una idea similar al algoritmo Shellsort, pero esta vez modficando el algoritmo Burbuja. El paso inicial (*gap*) se define como $\frac{n}{1{,}3}$, y se ordena los elementos separados por *gap* casilleros, partiendo desde el primero. Note que, a diferencia de Shellsort, sólo se realiza una pasada y luego se disminuye inmediatamente el gap por el factor 1,3. Cuando $gap = 1$ el algoritmo realiza tantas pasadas como sea necesario para obetener el arreglo ordenado (es decir, a partir de este punto es equivalente al algoritmo Burbuja, pero con todo el trabajo realizado antes debería tomarle mucho menos tiempo terminar de ordenar el arreglo).

Comparación de los algoritmos

Los métodos de ordenación deben generar conjuntos ordenados de menor a mayor.

La característica que se va a medir es contar el número de comparaciones entre pares de elementos del conjunto a ordenar, por lo cual tienen que contemplar este requerimiento en la implementación de los algoritmos de ordenación.

Luego tienen que realizar tres tipos de pruebas de ordenación:

1. Un conjunto con elementos generados aleatoriamente (que por supuesto es un conjunto desordenado).

2. El conjunto resultante de la Prueba 1, con elementos ordenados de menor a mayor.

3. El conjunto resultante de la Prueba 1, con los elementos ordenados de mayor a menor.

El tamaño de los conjuntos debe variar para cada tipo de prueba. Consideren conjuntos de tamaño 100.000, 200.000, 300.000, 400.000, 500.000, 600.000, 700.000, 800.000, 900.000 y 1.000.000. Consideren elementos enteros en el rango [0, 10.000.000]. Deben tener en cuenta que cuando generan elementos aleatorios puede ocurrir que tengan elementos repetidos, por lo que tienen que adecuar el algoritmo para soportar el caso que se repitan elementos de modo de minimizar las comparaciones.

El programa debe generar tres archivos de texto (insercion.txt, burbuja.txt y shellsort.txt) donde se muestren los resultados obtenidos según el siguiente formato:

```
<numElem> <numCompAleat>  <numCompMenorMayor>  <numCompMayorMenor>
```

Ejemplo:

```
burbuja.txt
100000  <numCompAleat>  <numCompMenorMayor>  <numCompMayorMenor>
200000  <numCompAleat>  <numCompMenorMayor>  <numCompMayorMenor>
...
1000000 <numCompAleat>  <numCompMenorMayor>  <numCompMayorMenor>
```

En el informe tienen que presentar en tres tablas los resultados obtenidos en las distintas pruebas (generados aleatoriamente, ordenados de menor a mayor, ordenados de mayor a menor), y adjuntar un gráfico asociado a cada tabla. De este modo podrán comparar los tres algoritmos para cada tipo de prueba.

Por último, deben realizar el ajuste polinomial que corresponda a cada prueba para cada algoritmo (son 9 ajustes en total), y deben entregar una pequeña conclusión sobre qué algoritmo es mejor para cada caso (generados aleatoriamente, ordenados de menor a mayor, ordenados de mayor a menor).

Restricciones

- El programa debe ser escrito en Java. No se acepta ningún otro lenguaje de programación.
- Plazo de entrega: 2 semanas desde la fecha de publicación (vean el encabezado). NO SE ACEPTARAN TAREAS ATRASADAS.
- Queda prohibido utilizar bibliotecas o clases preconstruidas en el lenguaje que implementen:

    - Listas, o arreglos (por ejemplo NO use la clase List y sus derivadas, y NO use la clase Arrays y sus derivadas).
    - Algoritmos de ordenación solicitados.

- La tarea es individual.
- Tienen que respetar los nombres de los archivos y los formatos de estos según las indicaciones entregadas en el enunciado. Se evaluará el cumplimiento de esta norma.
- Para la generación de elementos aleatorios pueden utilizar el método Math.random(), o un objeto de la clase Random (mayor referencia en la API de Java). Se recomienda el uso de la clase Random.
- Para la tarea tienen que entregar un informe de resultados junto al código fuente (el programa que implementa la tarea). No se aceptarán códigos sin informe ni informes sin código.
- SOLO SE ACEPTARAN TAREAS ENTREGADAS A TRAVES DE U-CURSOS.

Cualquier falta a las restricciones anteriores invalida irrevocablemente la tarea.

Ponderación

El informe equivale a un 50 % de la nota de tarea.
El código equivale a un 50 % de la nota de tarea.

Entrega de la tarea

La entrega de la tarea debe incluir un informe donde se describa y documente el programa realizado, y se incluyan ejemplos de entradas y salidas.

El informe (en formato PDF) y el código fuente (junto a un archivo README que explique como compilar) deben ser entregado a través del sistema U-Cursos hasta las 23:59 de la fecha de entrega.

El contenido del informe debe seguir la siguiente pauta (común para todas las tareas del Semestre):

- Portada:

  Universidad de Chile

  Facultad de Ciencias Físicas y Matemáticas

  Departamento de Ciencias de la Computación

  Informe Tarea X

  Nombre de la tarea X

  Fecha: fecha entrega

  Autor: nombre alumno

  E-mail: correo del alumno

- Introducción: Descripción del problema y de su solución.

- Análisis del problema: Se expone en detalle el problema, los supuestos que se van a ocupar, las situaciones de borde, y la metodología para abordar el problema.

- Solución del problema:

  - Algoritmo de solución: Se detallan los pasos a seguir para solucionar el problema. De ser necesario, se recomienda incluir figuras, tablas, etc., que permitan mejorar la comprensión del problema.
  - Diseño: Mostrar los invariantes empleados.
  - Implementación: Se debe mostrar el código del programa que soluciona el problema (omita los detalles que no tengan relevancia para el problema), explicando lo que hace el código. Se recomienda usar nombres representativos a las variables. NOTA: El código generado para resolver la tarea debe corresponder al diseño descrito, preocúpese de realizar los comentarios que sean necesarios.
  - Modo de uso: Se debe explicar el modo de uso del programa y el modo de compilación. Nota: si las indicaciones que Usted entrega son erróneas al momento de compilar, la evaluación de su tarea será severamente penalizada.

- Resultados: En esta sección se deben indicar las pruebas realizadas y sus resultados. Se debe señalar claramente cuál es la llamada al programa y su salida. En esta sección debe incluir las tablas y gráficos necesarios solicitados, y el análisis de los resultados. En este capítulo adjunten las conclusiones obtenidas de los resultados de la tarea.

- Anexos:
  - Listados: Se debe incluir un listado con el programa completo que se compiló realmente (utilicen una fuente pequeña (8 pt) en este listado).
  - Otros: De ser necesario, cualquier información adicional se debe agregar en los anexos y debe ser referenciada en alguna sección del informe de la tarea.