UNIVERSIDAD DE COSTA RICA

Escuela de Ciencias de la Computación e Informática

CI-0116 Análisis de Algoritmos y Estructuras de Datos

Tarea I

1. Objetivo

El objetivo de la tarea es implementar los algoritmos de ordenamiento vistos en el curso y comprobar que sus diferencias en cuanto a eficiencia teórica corresponden con la realidad.

2. Algoritmos

Los algoritmos a implementar son los siguientes: (I) ordenamiento por selección [8 pts.], (II) ordenamiento por inserción [8 pts.] y (III) ordenamiento por mezcla (*merge sort*) [8 pts.] para la **primera parte**, y (IV) ordenamiento por montículos (*heapsort*) [8 pts.], (V) ordenamiento rápido (*quicksort*) [8 pts.] y (VI) ordenamiento por residuos (*radix sort*) con dígitos en base $2^{\lfloor \lg n \rfloor}$ [15 pts.] para la **segunda parte**.

3. Comparación

Para comparar los algoritmos realice lo siguiente:

1. Cree arreglos de enteros (positivos y negativos) seleccionados al azar, de tamaño 50 000, 100 000, 150 000, y 200 000 y ejecute cada uno de los algoritmos al menos tres veces con cada uno de los arreglos.[1] Registre en un cuadro cada uno de estos tiempos y su respectivo promedio. Refiérase a la variación de los tiempos en las tres corridas (es decir, si fueron similares, iguales, muy distintos, etc.). [2.5 pts. c/ algoritmo].

[1] Si el tiempo mayor es 1,5 veces más grande que el tiempo menor, se debe posiblemente a que la máquina estuvo ocupada en otras cosas mientras hizo el ordenamiento. En tal caso elimine una de las corridas y vuelva a ejecutarla. Repita esto cuantas veces sea necesario para lograr que el tiempo mayor no sea más grande que 1,5 veces el tiempo menor para cada algoritmo.

2. Grafique los tiempos promedio contra el tamaño del arreglo para cada uno de los algoritmos y analice la curva resultante para identificar si su forma es la esperada (es decir, si las curvas de los algoritmos $\Theta(n^2)$ son aproximadamente parabólicas y si las curvas de los algoritmos $\Theta(n)$ y $\Theta(n \log n)$ son aproximadamente lineales).[2] Asegúrese de indicar en cada gráfico las unidades de tiempo utilizadas: s, ms, etc. [2.5 pts. c/ algoritmo].

3. Grafique de nuevo los tiempos promedio de ejecución pero póngalos en un mismo par de ejes. Muestre el tiempo en escala logarítmica para contrarrestar las enormes diferencias que probablemente existan entre los tiempos de ejecución de los algoritmos cuadráticos y los aproximadamente lineales. Identifique si la relación entre las curvas fue la esperada [2.5 pts. c/ algoritmo].

4. Entregables

La tarea tiene dos entregas. La primera es un informe preliminar en formato PDF con un análisis de los algoritmos de ordenamiento por selección, inserción y mezcla, y la segunda un informe final con esos algoritmos y los cuatro restantes: ordenamiento por montículos, ordenamiento rápido y ordenamiento por residuos (con dígitos en base $2^{\lfloor \lg n \rfloor}$). A cada uno de los reportes debe adjuntarse el archivo «Ordenador.h» con los algoritmos implementados. (Impleméntelos ahí, no en un *cpp*).

El código debe escribirse en el lenguaje de programación C++, tomando como base la plantilla publicada junto a este enunciado. Los encabezados de las funciones en la plantilla *no deben ser alterados*, ya que los asistentes usarán un guion (*script*) para revisar la tarea y cualquier cambio en la interfaz de los métodos afectará la compilación. (Sin embargo, se pueden agregar métodos privados y llamarlos dentro de los métodos definidos en la plantilla si fuera necesario). Se debe usar el lenguaje en su versión estándar para evitar problemas de compilación. En particular, el código debe poder ser compilado usando `g++` y las librerías estándar de C++. *Si el código no compila o no ordena correctamente, recibirá una nota de cero.* Para determinar si el algoritmo ordena correctamente se correrá la siguiente prueba:

```
1 for(int i=1; i<n; i++)
2     if( A[i] < A[i-1] )
3         cout << "¡Falló!";
```

Si cometió errores en la primera parte de la tarea, se le insta a corregirlos en la segunda parte; sin embargo, no se le reconocerán de vuelta los puntos perdidos.

5. Forma de entrega

La tarea debe entregarse en un archivo comprimido mediante la plataforma Mediación Virtual. (La tarea se puede dejar en estado «borrador»; no es necesario «enviarla

[2] Recuerde que $n \log n < n^{1+\epsilon}$ para $\epsilon > 0$ y n suficientemente grande.

a revisión»). Es responsabilidad del estudiante verificar que la plataforma haya recibido la tarea y que esté intacta (bajando la tarea y verificando la integridad del comprimido). En caso de que haya múltiples entregas, se considerará solamente la *última*, y si esta se entregó después de la fecha y hora límites, se aplicará la penalización acordada en la Carta al Estudiante: *la calificación recibida por tareas tardías no será más alta que la calificación recibida por cualquiera de los estudiantes que haya entregado la tarea a tiempo.*

Si tiene problemas para subir la tarea y el plazo está cerca de cumplirse, envíela a los asistentes con copia al docente. Favor poner en el asunto «Tarea 1 de CI-0116». En caso de que se reciban múltiples versiones por correo electrónico, se revisará solamente la *primera* enviada (para desalentar el envío de múltiples correos electrónicos). La política de penalización por entrega tardía mencionada en el párrafo anterior también aplica a entregas por correo electrónico.