UNIVERSIDAD DE COSTA RICA

Escuela de Ciencias de la Computación e Informática

CI-0116 Análisis de Algoritmos y Estructuras de Datos

I ciclo 2024

Prof. Dr. Allan Berrocal Rojas

Tarea I

Índice

1. Indicaciones generales

Esta tarea es individual y debe entregarla en la fecha que se indica en la plataforma de Mediación Virtual. Si lo desea, el(la) estudiante puede colaborar con sus compañeros(as) pero de manera profesional. Es decir, antes de discutir los problemas de la tarea con sus compañeros(as), primero dedique al menos un día de trabajo individual y formule sus propias respuestas para la tarea. Aunque colabore con algún compañero(a), todo el material que desarrolla y entrega en su tarea debe ser hecho por usted mismo(a). Cada estudiante debe estar en capacidad de defender correctamente las respuestas de su trabajo de manera verbal si la persona docente o la persona asistente del curso formula una pregunta durante la revisión del trabajo. Si la persona docente sospecha que la tarea es parcial o totalmente producto de plagio, la tarea recibirá una nota de cero y se iniciará el debido proceso estipulado en el Reglamento de Orden y Disciplina de los Estudiantes Universidad de Costa Rica. En el reporte a entregar, debe aparecer la referencia toda fuente externa de producción intelectual que utilizó para durante el desarrollo de este trabajo. Recuerde que se considera plagio presentar como propio, material parcial o totalmente creado por otras personas u obtenido de fuentes de información, como por ejemplo de libros, fuentes de Internet, sistemas de generación de texto basados en modelos de aprendizaje automático y profundo, entre otras.

2. Objetivo

El objetivo de la tarea es implementar los algoritmos de ordenamiento vistos en el curso para realizar experimentos y recolectar información sobre el desempeño real de los algoritmos. Se espera que la persona estudiante sea capaz de analizar los resultados y reflexionar sobre la eficiencia teórica de los algoritmos y la eficiencia observada durante los experimentos.

3. Algoritmos a Implementar

Los algoritmos a implementar en la primera parte son los siguientes:

1. ordenamiento por selección (*selection sort*) **[8 pts.]**
2. ordenamiento por inserción (*insertion sort*) **[8 pts.]**
3. ordenamiento por mezcla (*merge sort*) **[8 pts.]**

Los algoritmos a implementar en la segunda parte son los siguientes:

4. ordenamiento por montículos (*heapsort*) **[8 pts.]**

5. ordenamiento rápido (*quicksort*) **[8 pts.]**
6. ordenamiento por residuos (*radix sort*) con dígitos en base $2^{\lfloor \lg n \rfloor}$ **[15 pts.]**

4. Código Fuente de los Algoritmos

El código debe escribirse en el lenguaje de programación C++, tomando como base la plantilla publicada junto a este enunciado «`Ordenador.h`». Los encabezados de las funciones en la plantilla ***no deben ser alterados***, ya que la persona asistente del curso usará 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 bibliotecas estándar de C++. *Si el código no compila o no ordena correctamente, recibirá una nota de cero.*

Agregue al archivo «`Ordenador.h`» una función con el nombre `ImprimirDatosDeTarea()` que devuelva una hilera con su número de carné, identificador de la tarea, e identificador de la etapa **[0.5 pts.]**. Por ejemplo, si su número de carné es B12345, esta función deberá devolver la siguiente cadena para la primera la primera entrega de la tarea 1: `b12345 Tarea 1 Etapa 1`.

Para determinar si el algoritmo ordena correctamente se correrá la siguiente prueba:

Listado de código 1: Ejemplo de código fuente

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

5. Recolección de Datos

Para recolectar datos que le permitan comparar los algoritmos realice, lo siguiente:

1. Cree los siguientes 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]. Capture el tiempo de duración en milisegundos y registre en un

[1] Preste atención al programa para que las tres ejecuciones de cada algoritmo reciban exactamente el mismo arreglo de entrada. Evite que la segunda y tercera ejecución reciban el arreglo ya ordenado.

cuadro cada uno de estos tiempos y su respectivo promedio [2]. Para cada algoritmo, observe y escriba un comentario sobre la variación de los tiempos en las tres corridas (es decir, si fueron similares, iguales, muy distintos, etc.). Esta información la va a utilizar en el reporte escrito de la entrega [**2.5 pts. c/ algoritmo**].

2. Para cada uno de los algoritmos, cree un gráfico con los tiempos promedio contra el tamaño del arreglo. Agregue a cada gráfico la cota superior (notación O) y la cota inferior (notación Ω) correspondiente a cada algoritmo. 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).[3] Asegúrese de indicar en cada gráfico las unidades de tiempo utilizadas (milisegundos). [**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. Esta información la va utilizar en el reporte escrito de la entrega. [**2.5 pts. c/ algoritmo**].

6. Reporte de Resultados en PDF

Luego de ejecutar los algoritmos como se indicó en la sección anterior, usted debe discutir los resultados observados que ya tiene registrados en su hoja de datos.

Para escribir el reporte, debe utilizar como base la plantilla de artículos científicos de la IEEE. Para simplificar el proceso, el docente le brindará una plantilla que puede utilizar en **Microsoft Word** o en **LaTeX**.

El reporte escrito tiene un valor de [**10 pts.**] para la primera entrega y [**5 pts.**] para la segunda entrega. Algunos de los elementos a revisar son: redacción, ortografía, cumplimiento de las secciones solicitadas, calidad y correctitud de los gráficos generados, claridad del análisis, profundidad, pertinencia y correctitud del análisis, integridad del contenido (correcto uso de referencias a material

[2] 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.

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

utilizado), correcto uso de la nomenclatura aplicable.

7. Descripción del Producto a Entregar

La tarea tiene dos entregas. Cada entrega contiene un informe escrito en formato PDF y el código fuente de los algoritmos implementados.

En la primera entrega se deben incluir los algoritmos de ordenamiento por selección, inserción y mezcla. En la segunda se deben incluir los algoritmos restantes: ordenamiento por montículos, ordenamiento rápido y ordenamiento por residuos (con dígitos en base $2^{\lfloor \lg n \rfloor}$). En cada entrega debe adjuntarse el archivo «`Ordenador.h`» con los algoritmos implementados. (Impleméntelos ahí, no en un *cpp* aparte).

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

Entregue su trabajo en su repositorio privado de control de versiones en una estructura de directorios **exactamente** como se muestra a continuación: [**0.5 pts.**]

- `\tareas_programadas\tp1\entrega1`
  Coloque aquí la versión del código de la primera entrega así como el documento de reporte en formato PDF.
- `\tareas_programadas\tp1\entrega2`
  Coloque aquí la versión del código de la segunda entrega así como el documento de reporte en formato PDF.

Asegúrese de utilizar un archivo `.gitignore` para evitar subir al repositorio de control de versiones archivos temporales o autogenerados durarante el proceso de compilación. [**0.5 pts.**]

Es responsabilidad de la persona 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.

Si tiene dudas sobre cómo utilizar el repositorio de control de versiones, busque al asistente del curso o a la persona docente **con antelación** para recibir ayuda. No espere hasta el día de la entrega para solicitar ayuda porque corre el riesgo de que no se le pueda atender.

8. Valor de cada entrega

El sigueinte cuadro muestra un resumen de los elementos a evaluar en cada una de las dos entregas de la tarea.

Cuadro 1: Resumen de puntajes por entrega

Etapa	Elemento	Puntaje
	Implementación de ordenamiento por selección	8
	Implementación de ordenamiento por inserción	8
	Implementación de ordenamiento por mezcla	8
	Función `ImprimirDatosDeTarea()`	0.5
1	Captura de tiempo de ejecución de algoritmos	7.5
	Gráficos individuales de tiempo de ejecución	7.5
	Gráfico comparativo de tiempo de ejecución	7.5
	Reporte escrito	10
	Estructura de directorios	0.5
	Buen uso del repositorio y `.gitignore`	0.5
	Total	**58 pts.**
	Implementación de ordenamiento por montículos	8
	Implementación de ordenamiento por rápido	8
	Implementación de ordenamiento por residuos	15
	Función `ImprimirDatosDeTarea()`	0.5
2	Captura de tiempo de ejecución de algoritmos	7.5
	Gráficos individuales de tiempo de ejecución	7.5
	Gráfico comparativo de tiempo de ejecución	7.5
	Reporte escrito	5
	Estructura de directorios	0.5
	Buen uso del repositorio y `.gitignore`	0.5
	Total	**60 pts.**