EL4203 Programación Avanzada - Primavera 2024
Tarea N°3

Profesores de Cátedra: Ignacio Bugueño, Pablo Martin
Profesor Auxiliar: Esteban Muñoz
Ayudantes: Agustín González, Ignacio Romero, Diego Torreblanca
Plazo de entrega: Martes 19 de Noviembre, 23:59 hrs

Pregunta 1: Algoritmos de Ordenamiento

a. **Optimización de QuickSort con Mediana de Tres**
Implemente el algoritmo de ordenamiento `QuickSort` en Python con una variación que utilice el método de *mediana de tres* para elegir el pivote. Analice experimentalmente el impacto de esta optimización en términos de eficiencia para arreglos de tamaño creciente (por ejemplo, $n = 10^3, 10^4, 10^5$). Incluya una gráfica que muestre el tiempo de ejecución en función del tamaño del arreglo.

b. **Radix Sort para Datos Alfanuméricos**
Diseñe e implemente una versión de `Radix Sort` capaz de ordenar un conjunto de datos alfanuméricos (strings) de diferente longitud. La implementación debe ser capaz de ordenar strings en orden lexicográfico. Pruebe su implementación con una lista de al menos 10,000 elementos alfanuméricos aleatorios y mida su desempeño.

c. **CombSort**
Investigue y luego implemente el algoritmo de ordenamiento `CombSort` en Python. Este algoritmo es una mejora de `Bubble Sort` que usa el concepto de *gap shrinking*. Justifique en qué casos su implementación es más eficiente que `Bubble Sort` y en cuáles su rendimiento no mejora significativamente. Compare los tiempos de ejecución entre ambos algoritmos usando un conjunto de datos de prueba de al menos 1,000 elementos aleatorios.

Pregunta 2: Funciones de Hash

a. **Implementación de una Tabla Hash con Resolución de Colisiones por Doble Hashing**
Implemente una tabla hash que utilice *doble hashing* para resolver colisiones. Asegúrese de que la tabla soporte inserciones, búsquedas y eliminaciones, manejando correctamente las colisiones. Elija una función de hash principal y una función de hash secundaria que minimicen las colisiones. Evalúe la eficiencia de su implementación comparándola con una implementación de tabla hash con resolución de colisiones por *encadenamiento*.

b. **Función de Hash para Comparación de Textos**
Implemente una función de hash para detectar cadenas de texto duplicadas en un archivo grande. La función debe ser capaz de procesar archivos de texto de al menos 1 MB y retornar todas las líneas duplicadas. Considere usar el algoritmo `Rolling Hash` para reducir el tiempo de procesamiento y el uso de memoria. Comente sobre la eficiencia de su implementación y cómo se compara con el uso de una función de hash simple.

c. **Hash de Conteo de Elementos con Alta Frecuencia**
Diseñe un programa que permita identificar los elementos que aparecen con mayor frecuencia en una lista de elementos. Use una función de hash que maximice la eficiencia de las búsquedas y permita manejar grandes volúmenes de datos (al menos 100,000 elementos). Compare el rendimiento de su función con el método `counter` de la biblioteca estándar `collections` y analice las ventajas y desventajas de su enfoque.

Instrucciones

- No se aceptan tareas atrasadas.
- La implementación y el informe tendrán notas separadas, y estas se ponderaran en un 50% y un 50% respectivamente.
- Entregar el código (usando el lenguaje de programación `Python`) y el informe (.pdf) en la sección tareas de U-Cursos.
- Realizar preguntas sobre la tarea solamente por foro de U-Cursos.

Consideraciones

Entregable: El informe y códigos deben ser subidos a U-Cursos a más tardar el día señalado en el enunciado. En su entrega final, debe incluir un breve comentario indicando cómo ejecutar su programa.

El informe debe contener como mínimo: resumen, introducción, metodología (problema a abordar, soluciones, partes relevantes del código), resultados, análisis, conclusiones generales. Cada elemento en el enunciado debe ser abordado en el informe. Su extensión máxima es de 10 páginas. Se les proveerá de una plantilla en LaTeX, a doble columna.

Importante: La evaluación de la tarea considerará el correcto funcionamiento del programa, la inclusión de los resultados de los pasos pedidos en el informe, la calidad de los experimentos realizados y de su análisis, la inclusión de las partes importantes del código en el informe, así como la forma, prolijidad y calidad del mismo.

Nota: Se realizará una revisión acuciosa para la detección de copias/plagios. Para evitar problemas, los estudiantes deben programar todo lo que se pide por sí mismos y escribir el informe en su totalidad. Las copias/plagios serán penalizadas(os).