TAREA EXTRACLASE 2

Análisis de Algoritmos

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computadores
Algoritmos y Estructuras de Datos II (CE 1103)
II Semestre 2024

OBJETIVO GENERAL

- Analizar los tiempos de ejecución empíricos con respecto a los tiempos de ejecución teóricos en las operaciones básicas de estructuras de datos y algoritmos de ordenamiento.

OBJETIVOS ESPECÍFICOS

- Desarrollar una aplicación en el lenguaje de programación C++.
- Implementar algoritmos de ordenamiento sobre arreglos.
- Implementar las estructuras de datos básicas junto con sus operaciones más comunes.
- Medir los tiempos de ejecución de las operaciones más comunes de las estructuras de datos básicas.
- Medir los tiempos de ejecución de los algoritmos de ordenamiento.
- Comparar los tiempos de ejecución obtenidos contra los teóricamente esperados.

DESCRIPCIÓN DEL PROBLEMA

Es importante cuando se diseñan algoritmos, realizar el proceso de análisis de complejidad. Dicho proceso trata de inferir el comportamiento de un algoritmo conforme cambia el tamaño de su entrada sin tomar en cuenta el ambiente en el cual se encuentra ejecutando (PC, cantidad memoria, sistema operativo, etc). El análisis de complejidad permite asociar una función matemática con un algoritmo. Por ejemplo:

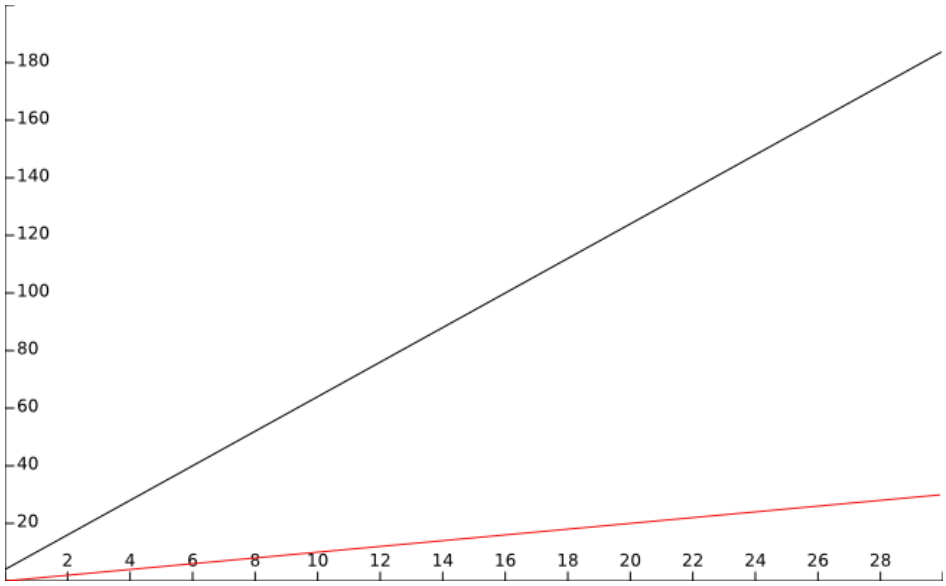
```java
boolean search(int arr[], int n, int num)
{
    boolean flag = false;
    for (int i = 0; i < n; i++)
    {
        if (arr[i] == num)

        {
            flag = true;
            break;
        }
    }
}
```

Al algoritmo anterior se le asocia la función f(N)= 7NT + 8T, por lo que podemos decir que la complejidad de este algoritmo es O(N).

Por otra parte, el análisis empírico o benchmark, consiste en realizar una medición del tiempo de ejecución modificando el tamaño de la entrada y capturando el tiempo que está tardó esta, de la siguiente forma:

```
tiempo1 = getSystemTimeNano();
llamarMetodo(N);
tiempo2 =  getSystemTimeNano();
tiempoTardado = tiempo2 - tiempo1;
```

Si realizamos este benchmark contra nuestro algoritmo del cual conocemos que su tiempo de ejecución se comporta de manera lineal y realizamos esta prueba muchas veces con diferentes valores de N, deberíamos poder graficar estos tiempoTardado para sus diferentes valores de N y de esta manera obtener una gráfica que se comporte de forma lineal, demostrando que efectivamente el análisis de complejidad refleja el comportamiento real observado en el benchmark.



La siguiente tabla presenta el análisis de complejidad para los algoritmos más comunes:

Algoritmo	Peor Caso	Mejor Caso	Caso promedio
BubbleSort	O(n^2)	O(n)	O(n^2)
SelectionSort	O(n^2)	O(n^2)	O(n^2)
MergeSort	O(n log n)	O(n log n)	O(n log n)
All Sorted LinkedList Search	O(n)	O(1)	O(n)
Binary Search Tree Insert	O(n)	O(log n)	O(log n)

Se solicita lo siguiente:

1. Implementar las diferentes estructuras y algoritmos de manera genérica en C++. Por simplicidad, utilizaremos únicamente enteros para realizar las pruebas.
2. Generar un programa de prueba que para cada operación en las estructuras de datos y para cada algoritmo de la tabla anterior, generar diferentes cantidades de datos de prueba (diferentes valores de N) que representan el peor de los casos, el mejor y el caso promedio, medir la duración de la ejecución del

algoritmo/operación con estos datos de prueba y graficar este conforme crece N. En esta gráfica deberá aparecer también la gráfica del valor teórico para poder comparar la tasa de crecimiento obtenida versus la tasa de crecimiento esperada.

ASPECTOS OPERATIVOS

- El trabajo se realizará de forma **individual**
- El uso de Git y Github es obligatorio
- La fecha de entrega será según lo especificado en el TEC Digital. Se entrega en el TEC digital, un archivo .txt dentro del cual hay un URL del repositorio donde se encuentra el código.
- El repositorio en Github debe tener un README que indique cómo se ejecuta la aplicación y cómo se usa.
- Las gráficas deben dibujarse en C++ con GTK o QT.