Logotipo

Descripción generada automáticamente

Algoritmos y Estructuras de Datos / Algoritmos y Programación 2

**INFORME DE COMPLEJIDAD ALGORÍTMICA**

Nombre del grupo: NOMBRE\_GRUPO

Integrantes:

* Candela Rivero Ardissone - 110339
* Moyano Andrés - 110017
* Camilo D. Campos - 109368
* Steven Guerrero - 110067
* Matías Tomás Petcoff - 111498

**Resumen**

Este informe presenta un análisis de la complejidad algorítmica del heap, enfocándose en las operaciones de "upheap" y "downheap". Mediante pruebas experimentales, se midió el promedio de estas operaciones en heaps de diferentes tamaños (100, 1000, y 10000 elementos). Los resultados fueron graficados y comparados con las expectativas teóricas, mostrando la eficiencia de las estructuras y algoritmos utilizados.

**Introducción**

En estructuras de datos, los heaps son utilizados comúnmente para implementar colas de prioridad. El heap garantiza que la operación de extracción del elemento máximo (o mínimo, dependiendo del tipo de heap) y la inserción se realicen en tiempo logarítmico respecto al número de elementos. La eficiencia de estas operaciones es crítica para aplicaciones de tiempo real y sistemas de alta performance.

Este informe tiene como objetivo analizar la complejidad temporal de las operaciones de inserción y extracción en un heap, específicamente midiendo las cantidades de "upheaps" y "downheaps" requeridas durante estas operaciones. Se evaluaron heaps de 100, 1000 y 10000 elementos, utilizando valores aleatorios para las inserciones.

**Desarrollo**

Para analizar la complejidad, se implementaron métodos en la clase heap que cuentan las operaciones de "upheap" y "downheap". Los métodos alta\_complejidad y baja\_complejidad fueron creados para este propósito. Se realizaron 100 pruebas de inserción y extracción para cada tamaño de heap, registrando el número de operaciones necesarias.

**Código**

El siguiente código muestra la implementación de los métodos de conteo de operaciones:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

El siguiente código muestra la realización de las pruebas:

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media  
Texto

Descripción generada automáticamente

**Resultados**

A continuación, se presentan los resultados obtenidos:

* **Promedio de upheaps (8 veces):**
* Para tamaño 100: 1.07, 0.92, 1.04, 1.01, 1.29, 1.23, 1.15, 1.03

Imagen que contiene biombo, mucho, grupo, parado

Descripción generada automáticamente

* Para tamaño 1000: 1.219, 1.231, 1.297, 1.26, 1.292, 1.22, 1.261, 1.169

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

* Para tamaño 10000: 1.2452, 1.2783, 1.2826, 1.277, 1.2718, 1.3081, 1.2642, 1.2787

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

* **Promedio de downheaps(8 veces):**
* Para tamaño 100: 5.07, 5.09, 5.24, 5.16, 5.13, 5.07, 5.06, 5.19

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

* Para tamaño 1000: 9.66, 9.68, 9.56, 9.72, 9.69, 9.65, 9.72, 9.7

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

* Para tamaño 10000: 13.05, 13.14, 13.24, 13.25, 13.04, 13.05, 13.01, 13.19

Gráfico

Descripción generada automáticamente

Estos resultados muestran un incremento gradual en el número de operaciones a medida que aumenta el tamaño del heap, lo cual es consistente con la complejidad teórica de O(log(n)).

En el análisis teórico, se espera que el número de operaciones "upheap" y "downheap" crezca logarítmicamente con respecto al tamaño del heap. Los resultados experimentales obtenidos confirman esta expectativa, demostrando la eficiencia de las estructuras y algoritmos implementados.

**Conclusiones**

El análisis realizado confirma que las operaciones de inserción y extracción en el heap se comportan de manera eficiente, cumpliendo con la complejidad temporal esperada de O(log(n)). Esto responde al uso de heaps en aplicaciones donde la eficiencia temporal es crucial. Las implementaciones de los métodos alta\_complejidad y baja\_complejidad proporcionan una herramienta útil para evaluar y optimizar el rendimiento de los algoritmos utilizados.

El presente informe demuestra, a través de pruebas empíricas, que la estructura de datos heap es altamente eficiente para las operaciones de inserción y extracción, y reafirma su idoneidad para su uso en contextos de alta demanda computacional.