Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Campus Sonora Norte

Act 3.4 - Actividad Integral de Árboles (Evidencia Competencia)

Curso:

Programación de estructuras de datos y algoritmos fundamentales

(TC1031)

Estudiante:

Daniel Alfredo Barreras Meraz A01254805

Docente:

Baldomero Olvera Villanueva

Fecha de entrega:

14 de octubre de 2023

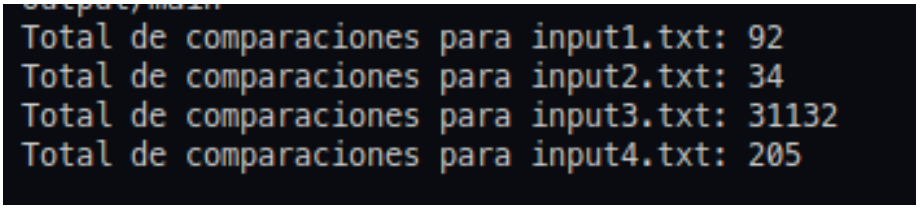
Código fuente

- Adjunto en el archivo zip.

Casos de prueba

Caso de prueba	**Archivo**	**Entrada**	**Resultado**
1	push()	5 20, 5, 8, 7, 4	92
2	pop()	7 1, 1, 1, 1 3, 3, 3	32
3	top()	134 1, 2, 1, 1, 2 6, 6, 4, 5, 6 11, 12, 4, 10, 7 6, 2, 1, 1, 7 9, 5, 9, 2, 8 21, 17, 22, 10, 25 9, 6, 4, 23, 15 35, 25, 5, 20, 13 7, 15, 39, 38, 39 5, 34, 37, 17, 23 46, 48, 2, 45 21,19 ,56 ,18 ,36 51 ,57 ,39 ,51 ,28 64 ,23 ,55 ,4 ,6 65 ,53 ,53 ,72 ,41 70 ,75 ,70 ,4 ,73 35 ,57 ,69 ,72 ,65 35 ,85 ,37 ,35 ,42 35 ,30 ,16 ,13 35 ,39 ,50 87 ,27 84 16 44 12 16 21 63 76 39 51 55 59 29 65 30 88 68 33 85 80 111 2 90 16 43 120 48 17 69 71 58 49 84 96 12 67	31132
4	size()	20 1, 1, 1, 1 2, 2, 2 3, 3, 3 4, 4, 4 5, 5, 5 1, 1, 1, 1	205

Comprobación de casos de prueba



Complejidad temporal:

push: O(log n) - Esta función añade un valor al final del array que representa el heap. Luego, reestructura o "heapifica" el heap para asegurar que el nodo con el valor más grande sea la raíz del árbol. Este proceso de reestructuración implica intercambiar nodos y puede requerir atravesar la altura del árbol, lo cual da una complejidad de tiempo de O(log n).

pop: O(log n) - Esta función elimina el nodo raíz (el valor más grande) del heap, lo reemplaza con el último nodo en el heap, y luego reestructura o "heapifica" el heap para asegurar que el nodo con el valor más grande sea la raíz del árbol. Al igual que con la función push, este proceso de reestructuración puede requerir atravesar la altura del árbol, lo cual da una complejidad de tiempo de O(log n).

top: O(1) - Esta función simplemente devuelve el valor del nodo raíz del heap, lo cual es una operación constante y por lo tanto tiene una complejidad de tiempo de O(1).

empty: O(1) - Esta función verifica si el heap está vacío comprobando si el tamaño del heap es cero. Como el tamaño del heap se almacena como una variable en la cola de prioridad, esta comprobación es una operación constante y por lo tanto tiene una complejidad de tiempo de O(1).

size: O(1) - Esta función devuelve el tamaño del heap, que se almacena como una variable en la cola de prioridad. Por lo tanto, es una operación constante y tiene una complejidad de tiempo de O(1).