Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Campus Sonora Norte

Act 3.2 - Árbol Heap: Implementando una fila priorizada

Curso:

Programación de estructuras de datos y algoritmos fundamentales

(TC1031)

Estudiante:

Daniel Alfredo Barreras Meraz A01254805

Docente:

Baldomero Olvera Villanueva

Fecha de entrega:

12 de octubre de 2023

Código fuente

- Adjunto en el archivo zip.

Casos de prueba

Fila inicial:

85 45 30 17 15 19 22 8 9

Caso de prueba	**Función**	**Entrada**	**Fila antes**	**Resultado**
1	push()	516	85 45 30 17 15 19 22 8 9	516 85 30 17 45 19 22 8 9 15
2	pop()	3	516 85 30 17 45 19 22 8 9 15	85 45 30 17 15 19 22 8 9
3	top()	3	85 45 30 17 15 19 22 8 9	85
4	size()	3	85 45 30 17 15 19 22 8 9	9
5	empty()		85 45 30 17 15 19 22 8 9	Falso

Comprobación de casos de prueba

```
output/main
Fila incial: 85 45 30 17 15 19 22 8 9
-----------------------------------------
Prueba 1
Funcion: push
Input: 516
Fila antes: 85 45 30 17 15 19 22 8 9
Fila despues: 516 85 30 17 45 19 22 8 9 15
-----------------------------------------
Prueba 2
Funcion: pop
Fila antes: 516 85 30 17 45 19 22 8 9 15
Fila despues: 85 45 30 17 15 19 22 8 9
-----------------------------------------
Prueba 3
Funcion: top
Fila: 85 45 30 17 15 19 22 8 9
Top: 85
-----------------------------------------
Prueba 4
Funcion: empty y size
Fila: 85 45 30 17 15 19 22 8 9
Size: 9
Esta vacia?: false
-----------------------------------------
```

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).