Tarea 8 - Programación y Algoritmos 1

Juan Luis Baldelomar Cabrera y Abdiel Beltrán

October 12, 2020

1. Complejidad de Algoritmos Implementados para TernaryHeap.

   Solución. ***BottomUpHeapify***. *Por como está definido este algoritmo, su complejidad depende de la altura del árbol. Al ser un árbol ternario completo, su altura es*

   $$\lceil log_3(n) \rceil$$

   donde n es la cantidad de datos en el árbol. Por lo tanto la complejidad del algoritmo es

   $$O(\lceil log_3(n) \rceil)$$

   TopDownHeapify. *Este algoritmo también depende de la altura del árbol, por lo tanto su complejidad es*

   $$O(\lceil log_3(n) \rceil)$$

   Insert. *La carga de este algoritmo recae después de haber insertado el dato, cuando se verifica que la heap property se siga cumpliendo, y para ello se debe hacer una llamada al método BottomUpHeapify, por lo tanto su complejidad es*

   $$O(\lceil log_3(n) \rceil)$$

   .

   RemoveMax. *De igual manera, la carga de este algoritmo se encuentra después de haber eliminado el dato en donde se verifica que se siga cumpliendo la propiedad de un heap y se llama al método TopDownHeapify, por lo cual su complejidad es*

   $$O(\lceil log_3(n) \rceil)$$

   getMax. *Obtener el máximo es una operación de tiempo constante debido a que el máximo se encuentra en la posición 0 del arreglo que contiene los datos, por lo tanto su complejidad es de orden constante.*

   Podemos ver entonces que en comparación con el binary heap, es de menor complejidad ya que la altura del árbol ternario será menor. Sin embargo, sabemos que independientemente de la base, las funciones de orden logaritmico son del mismo orden en la notación O (probado en clase).

2. Demostrar que el algoritmo propuesto calcula bien el valor de la mediana

Demostración. *Sea H_M un max heap y h_m un min heap y sea μ la mediana antes de insertar el nuevo dato.*

Caso 1. *Procedamos primero a mostrar el caso más fácil, cuando tanto H_M y h_m tienen la misma cantidad de datos. Supongamos que tanto H_M como h_m tienen k datos cada uno. Entonces para todo $M \in H_M$ se cumple que*

$$M \leq \mu.$$

De manera análoga, para todo $m \in h_m$ se cumple

$$\mu \leq m.$$

Puesto que ambas estructuras tienen la misma cantidad de datos, si el nuevo dato a insertar es mayor que μ, entonces al insertar el nuevo dato en h_m se tiene que todo $m \in h_m$ cumple

$$\mu \leq m$$

y este heap tiene ahora $k+1$ datos y al obtener el mínimo de ellos m_0 obtenemos que para $\forall m \in h_m,\ m \neq m_0\ y\ \forall M \in H_M$,

$$M \leq \mu \leq m_0 \leq m$$

y se cumple que

$$|H_M| = |h_m \setminus m_0|$$

de manera que m_0 es la mediana.

Caso 2. *Ahora probemos el caso en que H_M tiene más elementos que h_m. Si el nuevo dato a insertar cumple con que $d < \mu$ entonces removemos el máximo de H_M, que de hecho es la mediana μ, y lo insertamos en h_m. Luego insertamos el nuevo dato en H_M. Ambas estructuras tienen la misma cantidad de datos y cumplen con que para todo $M \in H_m$ y para todo $m \in h_m$,*

$$M \leq \mu \leq m$$

.

Si llamamos a m_1 al mínimo de h_m y M_1 el máximo de H_M, entonces para todo $M \in H_M$ y para todo $m \in h_m$ tenemos que

$$M \leq M_1 \leq m_1 \leq m$$

lo cual implica

$$2M \leq M_1 + m_1 \leq 2m$$
$$\Rightarrow M \leq \frac{M_1 + m_1}{2} \leq m$$

y como $|H_m| = |h_m|$, en efecto

$$\frac{M_1 + m_1}{2}$$

es la mediana.

El caso en que el nuevo dato es mayor que la mediana, podemos ver que al insertarlo en h_m se cumple que $|H_M| = |h_m|$ y las desigualdades de arriba siguen cumpliendo, por lo cual el cálculo de la mediana de esta forma sigue estando bien definido.

Caso 3. *Ahora probemos el caso en que h_m tiene más elementos que H_M. Si el nuevo dato a insertar cumple con que $d > \mu$ entonces removemos el mínimo de h_m, que de hecho es la mediana μ, y lo insertamos en H_M. Luego insertamos el nuevo dato en h_m. Ambas estructuras tienen la misma cantidad de datos y cumplen con que para todo $M \in H_m$ y para todo $m \in h_m$,*

$$M \leq \mu \leq m$$

.

Si llamamos a m_1 al mínimo de h_m y M_1 el máximo de H_M, entonces para todo $M \in H_M$ y para todo $m \in h_m$ tenemos que

$$M \leq M_1 \leq m_1 \leq m$$

lo cual implica

$$\begin{aligned} 2M &\leq M_1 + m_1 \leq 2m \\ \Rightarrow M &\leq \frac{M_1 + m_1}{2} \leq m \end{aligned}$$

y como $|H_m| = |h_m|$, en efecto

$$\frac{M_1 + m_1}{2}$$

es la mediana.

El caso en que el nuevo dato es menor que la mediana, podemos ver que al insertarlo en H_M se cumple que $|H_M| = |h_m|$ y las desigualdades de arriba siguen cumpliendo, por lo cual el cálculo de la mediana de esta forma sigue estando bien definido.

■

3. Complejidad de cálculo de la mediana

Solución. *Podemos notar por como está definido el algoritmo que el peor escenario es cuando el max heap y el min heap difieren en cantidad de datos por 1 unidad. En ese caso, siempre hay que hacer una operación de RemoveMax [RemoveMin], luego una operación para insertar este elemento removido en el otro heap y finalmente insertar el nuevo dato en el heap del cual se removió el dato. Entonces tenemos dos operaciones de inserción y una operación de eliminación. Además el tiempo de cálculo de la mediana actual que es de complejidad constante, por lo tanto la complejidad del cálculo de la mediana tras insertar un dato es*

$$O(3\log_3(n))$$