

Table of Contents

Question 1:

(a)

(b)

Question 2:

Question 3:

Question 4:

Question 1.

(a)

$$\begin{aligned}A(n) &= n^2 A(n-1) \\A(n-1) &= (n-1)^2 A(n-2) \\A(n-2) &= (n-2)^2 A(n-3) \\&\vdots \\A(2) &= (2)^2 A(1) \\A(1) &= 1\end{aligned}$$

So by **Repeated Back Substitution**, we have:

$$\begin{aligned}A(n) &= n^2(n-1)^2(n-2)^2 \cdots (2)^2(1)^2 \\&= (n(n-1)(n-2) \cdots (2)(1))^2 \\&= (n!)^2\end{aligned}$$

$$\therefore A(n) \in \Theta((n!)^2)$$

(b)

$$C(n) = \begin{cases} 8C\left\lfloor \frac{n}{2} \right\rfloor + 2n^3 + 4n & \text{if } n > 0 \\ 6 & \text{if } n = 0 \end{cases}$$

We can calculate $C(1)$ by plugging it in.

$$\begin{aligned} C(1) &= 8C\left\lfloor \frac{1}{2} \right\rfloor + 2(1)^3 + 4(1) \\ &= 8C(0) + 2 + 4 \\ &= 8(6) + 6 \\ &= 54 \end{aligned}$$

Now we can rewrite $C(n)$ to...

$$C(n) = \begin{cases} 8C\left\lfloor \frac{n}{2} \right\rfloor + 2n^3 + 4n & \text{if } n > 1 \\ 54 & \text{if } n = 1 \end{cases}$$

We can see that this satisfies the **Generalized Master Theorem**, where

$$a = 8 \qquad b = 2 \qquad c = 3 \qquad d = 54 \qquad f(n) = 2n^3 + 4n$$

So we know that $f(n) \in \Theta(n^3)$, and $\log_2 8 = 3$
 $\implies \log_b a = c$

By part (b) of **Generalized Master Theorem**, we have

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a} \log_b n) \\ &= \Theta(n^3 \log_2 n) \end{aligned}$$

Problem 2.

Suppose...

$$f(n) \in \mathcal{O}(g(n)) \qquad f(n) \geq 1 \qquad \log(g(n)) \geq 1 \qquad \forall n \in \mathbb{N}$$

$$\textbf{WTS: } \log(f(n)) \in \mathcal{O}(\log(g(n))) \qquad \textbf{OR} \\ \exists c' \in \mathbb{R}^+, \exists n'_0 \in \mathbb{N}, \forall n > n'_0 \implies \log(f(n)) \leq c' \cdot \log(g(n)) \qquad (\star)$$

Since $f(n) \in \mathcal{O}(g(n))$ we have...

$$\exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n > n_0 \implies f(n) \leq c \cdot g(n)$$

$$\begin{aligned} f(n) &\leq c \cdot g(n) && \text{[by given]} \\ \log(f(n)) &\leq \log(c \cdot g(n)) && \text{[logging both sides since both sides } \geq 1] \\ &\leq \log(c) + \log(g(n)) && \text{[by log laws]} \\ &\leq \log(c) \log(g(n)) + \log(g(n)) && \text{[since } \log(g(n)) \geq 1] \\ &\leq (\log(c) + 1) \log(g(n)) && \text{[by factoring]} \end{aligned}$$

So choose:

$$c' = \log(c) + 1 \qquad n'_0 = n_0$$

Then the predicate (\star) holds ■

Problem 3.

Implementation:

for RECENT, I would use an AVL Tree and a Doubly Linked List.

I would also keep track of `size`, and the `head` and `tail` of the Doubly Linked List.

In the AVL Tree, the nodes would have an extra parameter:

`target`: a pointer to its node in the Doubly Linked List.

For every `ACCESS(x)`, it would insert `x` into both the AVL Tree and the Doubly Linked List.

If `x` is already in the AVL Tree, then use `target` to delete that node in the Doubly Linked List.

Then re-insert it back into the head of the Doubly Linked List by using `head`.

If `size > m`, then delete both `tail`, and its corresponding node in the AVL Tree. Then continue to insert `x`.

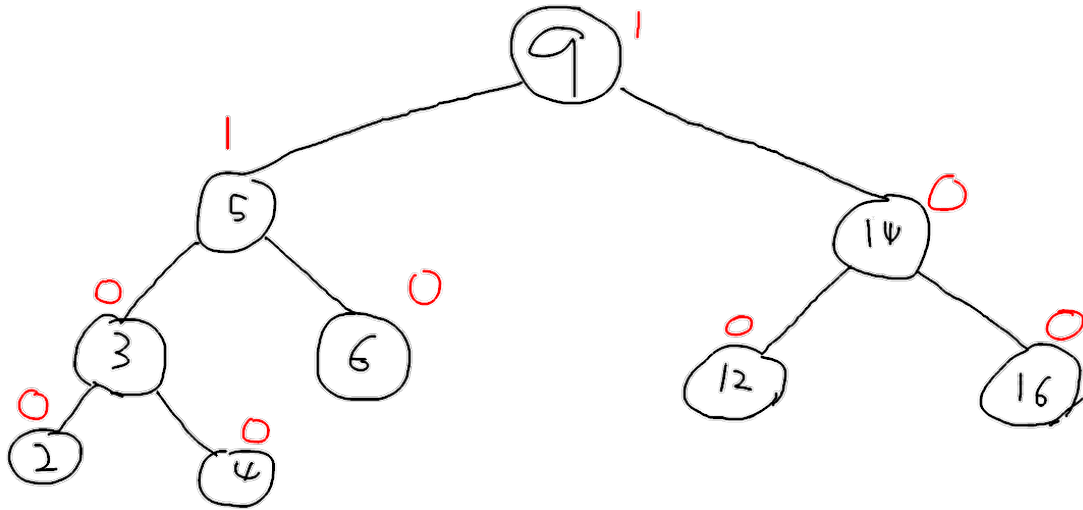
Justification:

This data structure uses $\mathcal{O}(m \log n)$ space since we have 2 data structures of size `m`. This means that we will only have about `2m` nodes to store, which according to this piazza post, takes about $2(m \log n) = \mathcal{O}(m \log n)$ space

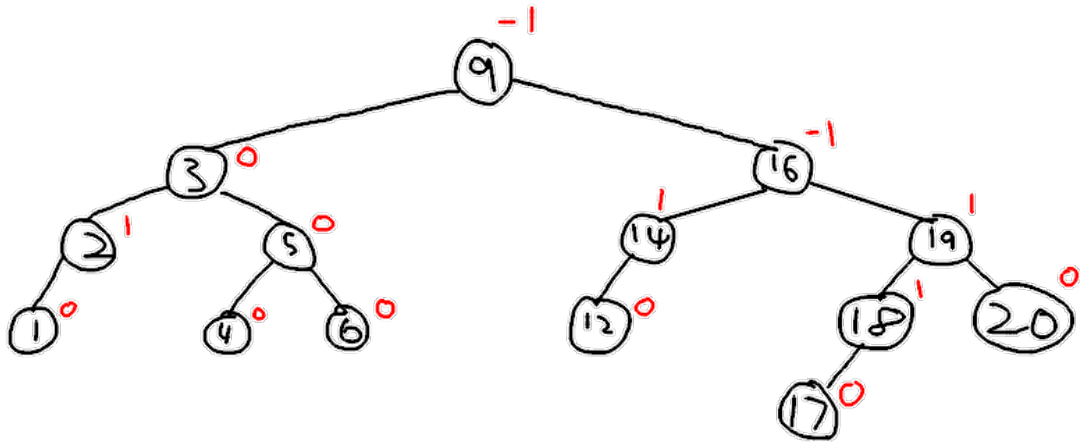
This data structure takes worst case $\mathcal{O}(\log m)$ time complexity. This is because for every `ACCESS(x)`, we need to search the AVL Tree which takes $\mathcal{O}(\log m)$ time. It could also delete and insert in the Doubly Linked List after searching, which would take $\mathcal{O}(1)$ since we keep a pointer to `size`, `head`, and `tail`. Lastly, we would also need to delete from both the AVL Tree and the Doubly Linked List if `size` gets too large, which would take $\mathcal{O}(\log m)$ and $\mathcal{O}(1)$ time respectively. Overall, the worst case time complexity is $\mathcal{O}(\log m)$.

Problem 4.

Tree after inserting 5, 4, 6, 9, 12, 16, 14, 2, 3



Tree after inserting 1, 20, 19, 18, 17



Tree after deleting 1, 2, 3, 4, 5, 12, 6

