Short Answers for Homework-2


The time cost is $O(\log n)$.


Notice if $A[n] < a$, then we can’t find the item. If $A[n] \geq a$, but for $i$ that satisfies and $2^i < n$ and $2^i + 1 \geq n$, we have $A[2^i] < a$, then do binary search between $A[2^i]$ and $A[n]$.

3. (b) If $\mu \rightarrow \text{pre} < \nu \rightarrow \text{pre}$, this means $\mu$ is a proper ancestor of $\nu$, or $\mu$ and $\nu$ share a common ancestor $\eta$, $\mu$ is in the left subtree of $\eta$ and $\nu$ is in the right subtree of $\eta$.

If $\mu \rightarrow \text{post} > \nu \rightarrow \text{post}$, this means $\mu$ is a proper ancestor of $\nu$, or $\mu$ and $\nu$ share a common ancestor $\eta$, $\mu$ is in the right subtree of $\eta$ and $\nu$ is in the left subtree of $\eta$.

If both conditions are satisfied, that means $\mu$ is a proper ancestor of $\nu$.

4. When a node is inserted into a red-black tree, the incoming edge is always red. Then we do rotations and promotions to keep the property of the red-black tree. However, all operation for insertion will keep at least one red edge in the tree.

5. When ENQUEUEing an element, put it into stack $A$. When DEQUEUEing an element, first check stack $B$. If stack $B$ is empty, pop all the elements in stack $A$ into stack $B$ and pop the top of stack $B$. If stack $B$ is not empty, pop the top of stack $B$.

Define the potential function $\Phi$ as $\Phi = 2 \times \text{size of stack } A + \text{size of stack } B = 2S_A + S_B$.

ENQUEUE:
Actual cost: $a_i = 1$
Amortized cost:

$$c_i = a_i + \Phi_i - \Phi_{i-1}$$

$$= 1 + (2(S_A + 1) + S_B) - (2S_A + S_B)$$

$$= 3$$

DEQUEUE:

1. When stack $B$ is empty:
  Actual cost: $a_i = S_A + 1$
Amortized cost:
\[
c_i = a_i + \Phi_i - \Phi_{i-1} \\
= (S_A + 1) + (2 \times 0 + (S_A - 1)) - (2S_A + 0) \\
= 0
\]

2. When stack \( B \) is not empty:
   - Actual cost: \( a_i = 1 \)
   - Amortized cost:
     \[
c_i = a_i + \Phi_i - \Phi_{i-1} \\
     = 1 + (2S_A + (S_B - 1)) - (2S_A + S_B) \\
     = 0
\]

So each \texttt{ENQUEUE} and \texttt{DEQUEUE} is \( O(1) \) amortized time.

6. \textbf{Bonus Problem:}

   Use one list \( A \) to store the element and another list \( B \) to store the minimum. Each element in \( A \) also has a pointer pointing to the corresponding element in \( B \) (if exists).

   \textbf{ENQUEUE:} Add the element \( a \) to the end of list \( A \). Compare this element \( a \) to the end of list \( B \), which is the element \( b \). If \( a > b \), add \( a \) directly to the end of list \( B \). If \( a < b \), remove \( b \) (the end of list \( B \)), and compare \( a \) to the new end of list \( B \). Keep doing this until the end of list \( B \) is smaller than \( a \), and put \( a \) at the end of list \( B \).

   \textbf{DEQUEUE:} Remove the head of list \( A \), and if there is a corresponding element in \( B \), remove it, too.

   \textbf{MINIMUM:} Return the head of list \( B \).

   Define the potential function \( \Phi \) as \( \Phi = \) length of stack \( A \) + length of stack \( B = L_A + L_B \).

   \textbf{ENQUEUE:} Suppose we remove \( k \) element from list \( B \)
   - Actual cost: \( a_i = 2 + k \)
   - Amortized cost:
     \[
c_i = a_i + \Phi_i - \Phi_{i-1} \\
     = (2 + k) + ((L_A + 1) + (L_B - k + 1)) - (L_A + L_B) \\
     = 4
\]

   \textbf{DEQUEUE:}

   1. When dequeuing the minimum:
      - Actual cost: \( a_i = 2 \)
      - Amortized cost:
        \[
c_i = a_i + \Phi_i - \Phi_{i-1} \\
        = 2 + ((L_A - 1) + (L_B - 1)) - (L_A + L_B) \\
        = 0
\]
1. When dequeuing the minimum:

   Actual cost: $a_i = 1$

   Amortized cost:

   $$c_i = a_i + \Phi_i - \Phi_{i-1}$$
   $$= 1 + ((L_A - 1) + L_B) - (L_A + L_B)$$
   $$= 0$$

**Minimum:**

   Actual cost: $a_i = 1$

   Amortized cost:

   $$c_i = a_i + \Phi_i - \Phi_{i-1}$$
   $$= 1 + (L_A + L_B) - (L_A + L_B)$$
   $$= 1$$

So all three operation are $O(1)$ amortized time.