Problem Set 3

Problem 3-1

(a) Answer: Min- and max-heaps enable constant-time retrieval of their min and max elements, respectively, but finding the maximum element in a min-heap or the minimum element in a max heap is $O(n)$. BSTs allow for $O(log_2 n)$ retrieval of both the minimum and maximum element, but only if they are balanced. In the case of extremely unbalanced trees, these operations are $O(n)$. AVL trees are balanced BSTs that enable retrieval of both minimum and maximum elements in $O(log_2 n)$ time, so the answer is AVL.

(b) Answer: Insertion into an AVL tree takes $O(log_2 n)$ time.

(c) Answer: Retrieving the minimum key an AVL tree takes $O(log_2 n)$ time.

(d) Answer: Retrieving the maximum key an AVL tree takes $O(log_2 n)$ time.

(e) Answer: $Rank(h) - Rank(l) + 1$. Just imagine an array $1, 2, 3, 4, 5$. Then $Count(3, 5) = 3$, $Rank(5) = 5$ and $Rank(3) = 3$, so $Count(3, 5) = Rank(5) - Rank(3) + 1$.

(f) Answer: $Rank(h) - Rank(l)$. If we remove 3 from the previous array, we have $1, 2, 4, 5$. Now $Count(3, 5) = 2$, $Rank(5) = 4$ and $Rank(3) = 2$, so $Count(3, 5) = Rank(5) - Rank(3)$.

(g). Answer: $Rank(h) - Rank(l)$. If we remove 5 from the original array, we have $1, 2, 3, 4$. Now $Count(3, 5) = 2$, $Rank(5) = 4$ and $Rank(3) = 3$, so $Count(3, 5) = Rank(5) - Rank(3) + 1$.

(h). Answer. $Rank(h) - Rank(l)$**.** If we remove both 3 and 5 from the original array, we end up with $1, 2, 4$. Now $Count(3, 5) = 1$, $Rank(5) = 3$ and 2, so $Count(3, 5) = Rank(5) - Rank(3)$.

(i). Answer. Choice 4, the number of nodes in the subtree rooted at $node$. This information can be updated in constant time as $node.left.\gamma + node.right.\gamma + 1$. In addition, this information can be used to calculate rank of $node$. To do so, initialize $rank = node.left.\gamma$ if $node.left$ exists, otherwise initialize $rank = 1$. Then travel up the three from $node$. Anytime you go left up the tree to a parent node $parent_L$, increment $rank$ by $parent.left.\gamma + 1$. Stop when you reach the root. This take time $O(log_2 n)$.

(j). Answer: You need $O(log_2 n)$ bits to store a number between 0 and n.

(k). Answer: A leaf node as $\gamma = 1$.

(l). Answer: $N_3.\gamma = 3$.

(m). Answer: $N_2.\gamma = 6$.

(n). Answer: $N_1.\gamma = 10$.