After Lecture 13 & 14 Practice Problems (all taken from previous exams)

1. If you want to create in order-statistic tree (which needs the size of each subtree rooted at each node), from an already created red-black tree, you can:

   a) perform a pre-order traversal of the order-statistic tree and sum the sizes of each subtree of a node and add one to get the size of each node (nodes with no children assigned size=1)

   b) perform an in-order traversal of the order-statistic tree and sum the sizes of each subtree of a node and add one to get the size of each node (nodes with no children assigned size=1)

   c) perform a post-order traversal of the order-statistic tree and sum the sizes of each subtree of a node and add one to get the size of each node (nodes with no children assigned size=1)

2. How does an augmented data structure differ from a traditional data structure?

   a) Augmented data structures have an asymptotically higher memory overhead.

   b) Augmented data structures worsen the asymptotic runtime of basic operations.

   c) Augmented data structures offer additional operations or information.

   d) Augmented data structures have a faster runtime complexity than the non-augmented data structure.

3. If a problem can be broken into sub-problems which are reused several times, the problem has ______.

   a) Overlapping subproblems

   b) Optimal substructure

   c) Memoization[1]

   d) Greedy

4. What is the space complexity of the dynamic programming implementation of the matrix chain problem?

   a) $O(1)$

   b) $O(n)$

   c) $O(n^2)$

   d) $O(n^3)$

5. Given an element x in an n-node order statistic tree and a natural number i, how can we determine the ith successor of x in the linear order of the tree in $O(\lg n)$ time? So x is a key in the tree and we want to find the ith key after x in linear order.

[1] **Memoization** means that we should never try to compute the solution to the

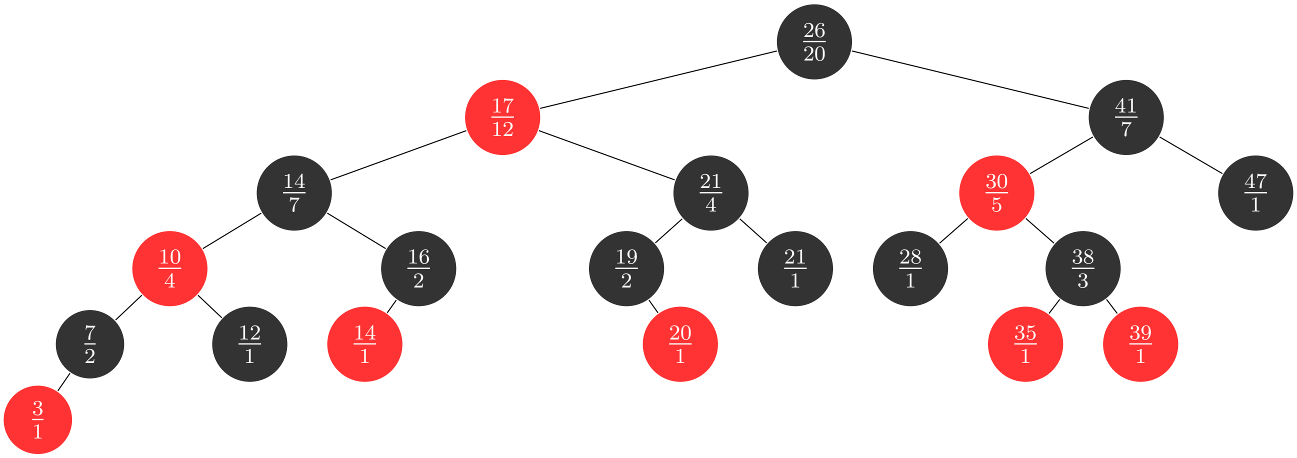

Figure 7.1: An order-statistic tree, which is an augmented red-black tree. In addition to its usual attributes, each node x has an attribute $x.size$, which is the number of nodes, other than the sentinel, in the subtree rooted at x.

First we determine the rank of x by calling OS-RANK(T, x) and name this number r. Then the ith successor of x is actually an element in the tree with rank $r+i$. Hence we call OS-SELECT($T.root$, $r+i$). Both these calls require $O(\lg n)$ time which in total is again $O(\lg n)$.

6. Suppose that the dimensions of the matrices A, B, C, and D are 8×5, 5×11, 11×6, and 6×9 respectively, and that we want to parenthesize the product $ABCD$ in a way that minimizes the number of scalar multiplications. Find the m and s tables computed by MATRIX-CHAIN-ORDER to solve this problem and show the optimal parenthesization.

Table 7.1:

m	A	B	C	D
A	0	440	570	960
B		0	330	600
B			0	594
B				0

7. Let $R(i,j)$ be the number of times that table entry $m[i,j]$ is referenced while computing other table entries in a call of MATRIX-CHAIN-ORDER. Show that the total number of references for the entire table is

$$\sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) = \frac{n^3-n}{3}$$

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) &= \sum_{l=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{i+l-2} 2 \\
&= \sum_{l=2}^{n}\sum_{i=1}^{n-l+1} 2(l-1) \\
&= \sum_{l=2}^{n} 2(l-1)(n-l+1) \\
&= \sum_{l=1}^{n-1} 2l(n-1)
\end{aligned}$$