Design and Analysis of Algorithms (CS345A)

Practice-sheet: Augmented Binary Search Trees

1. **Alternate solution for find-rank operation**

   In the class, we discussed the operation FIND-RANK(T, x) on a red-black tree. We found that we can perform this operation by augmenting each node v of the tree with a field SIZE(v) that stores the number of nodes in the subtree rooted at v. Can we keep any other field instead of SIZE field to solve this problem ? Find out which of the following fields we can keep and still achieve $O(\log n)$ time for each operation on red-black tree.

   - LEFT-SIZE(v): the size of the left subtree.
   - RANK(v): the rank of the element v in tree T.
   - SUBTREE-RANK(v): the rank of the element v in the subtree $T(v)$.

2. **Sequence of bits**

   Recall the XOR operation that you might have studied in Boolean arithmetic or electric circuits or elsewhere. Maintain a data structure for storing a sequence of n bits $\langle b_1, b_2, \cdots, b_n\rangle$ under the following operations.

   (a) Insert(i,b): Insert a bit at ith position in the sequence and set its value to b.

   (b) Delete(i): Delete the bit at ith position in the sequence.

   (c) ReportXOR(j,k): Return the XOR of bits $b_j, b_{j+1}, \ldots, b_k$ in the sequence.

   Each operation must take $O(\log n)$ worst case time.

3. **Intersecting chords**

   Given n chords in a circle, design an $O(n \log n)$ time algorithm to count their number of intersections.

4. **Application of line sweep method**

   In the class we discussed a line sweep method to solve the following problem. Given a set of n axis-parallel rectangles, determine whether there is any pair of them that intersect. The following is another interesting application of line sweep method.

   There are n horizontal line segments and each of them is colored red. There are n vertical line segments and each of them is colored blue. Design an $O(n \log n)$ time algorithm to count all the intersections between red and blue segments.

5. **Josephus problem**

   The Josephus problem is defined as follows. Suppose that n people are arranged in a circle and that we are given a positive integer $m \leq n$. Beginning with a designated first person, we proceed around the circle, removing every mth person. After each person is removed, counting continues around the circle that remains. This process continues until all n people have been removed. The order in which the people are removed from the circle defines the (n, m)-Josephus permutation of the integers $1, 2, ..., n$. For example, the $(7, 3)$-Josephus permutation is $\langle 3, 6, 2, 7, 5, 1, 4 \rangle$.

(a) Suppose that m is a constant. Describe an $O(n)$-time algorithm that, given an integer n, outputs the (n, m)-Josephus permutation.

(b) Suppose that m is not a constant. Describe an $O(n \log n)$ time algorithm that, given integers n and m, outputs the (n, m)-Josephus permutation.

1 Only for fun (not for exams)

1. **Sequence under rotations**

   Maintain a data structure for storing a sequence S of numbers $\langle a_1, a_2, \cdots, a_n \rangle$ under the following operations.

   (a) Insert(S,i,x): Insert a number x at position i in the sequence S.

   (b) Report(S, i): Return the value of ith number from the sequence S.

   (c) Delete(S, i): Delete ith number from the sequence S.

   (d) Rotate(S, i, j): Rotate the sequence from ith element to jth element. For example, if sequence is $S = \langle 3, 1, 66, 5, 9, 12, 34, 76 \rangle$, then after Rotate($S, 2, 5$), it becomes: $\langle 3, 12, 9, 5, 66, 1, 34, 76 \rangle$.

   Each operation must take $O(\log n)$ time.

2. **The mother of all data structures**

   This problem will test all your knowledge of binary search trees.

   Maintain a data structure for storing a sequence S of numbers $\langle a_1, a_2, \cdots, a_n \rangle$ under the following operations. Each operation must take $O(\log n)$ time.

   (a) Insert(S,i,x): Insert a number x at position i in the sequence S.

   (b) Delete(S, i): Delete ith number from the sequence S.

   (c) Rotate(S, i, j): Rotate the sequence from ith element to jth element. For example, if sequence is $S = \langle 3, 1, 66, 5, 9, 12, 34, 76 \rangle$, then after Rotate($S, 2, 5$), it becomes: $\langle 3, 12, 9, 5, 66, 1, 34, 76 \rangle$.

   (d) Multi-add(S, i, j, x): Add x to all numbers starting from the ith number and ending at jth number in the sequence S.

   (e) ReportMin(S, i, j): Return the smallest number from the ith number to jth number of the sequence S.