Advanced Data Structures (COP5536)

Solution - Exam 3, Sample 1

---

1. (a)

```
                        9
                  /           \
                 /             \
             3,6                12,15
          /   |   \           /     |      \
         /    |    \         /      |       \
      1,2   4,5   7,8    10,11   13,14   16,17
```

(b) Delete 3

Step 1: transform to leaf node deletion

```
                        9
                  /           \
                 /             \
             2,6                12,15
          /   |   \           /     |      \
         /    |    \         /      |       \
       1    4,5   7,8    10,11   13,14   16,17
```

Step 2: merge

```
                        9
                  /           \
                 /             \
               6                12,15
          /        \          /     |      \
         /          \        /      |       \
     1,2,4,5        7,8   10,11   13,14   16,17
```

Step 3: merge

```
                  6,   9,  12,  15
               /      /    |     \       \
              /      /     |      \       \
      /        /        |         \          \
1,2,4,5     7,8     10,11     13,14     16,17
```

2. Splay: delete 5

Step 1: Replace 5 with the largest element in its left subtree and, delete the node that had that largest element

```
      1
       \
        10
        /
       2
        \
         9
        /
       8
      /
     4
    / \
   3   6
        \
         7
```

Step 2 ... : Splay starting from the node with 4.

```
      1                        1                          4
       \                        \                       /     \
        10                       4                     1       10
        /                       / \                     \      /
       2              ==>      2   10        ==>         2    8
        \                       \  /                      \  / \
         4                       3 8                       3 6  9
        / \                       / \                        \
       3   8                     6   9                        7
          / \                     \
         6   9                     7
          \
           7
```

3. Step1:

```
                    12345
```

Step 2:

```
                  (4) [  4   7  ]
                     /         \
                    /           \
               12345           12374
```

Step 3:

```
                              (2) [  2   3 ]
                                 /           \
                                /             \
               (4) [  4   7  ]                 13344
                  /         \
                 /           \
             12345           12374
```

Step 4:

```
                              (2) [  2   3 ]
                                 /           \
                                /             \
               (4) [  4   7  ]                 (3) [  3   5  ]
                  /         \                     /          \
                 /           \                   /            \
             12345           12374           13344            13574
```

4. (10) For the min radix priority search tree (RPST) with the range [0, 32),

(a) Insertions

```
                                  (20,8)   [0,32)                     (1,4)   [0, 32)
                                     \                                     \
(23, 15)   [0, 32)      ==>         (21,10)   [16,32)        ==>          (20,8)   [16,32)
                                     /                                    /
                                  (23,15)   [16,24)                    (21,10)   [16,24)
                                                                            \
                                                                           (23,15)   [20,24)
```

(b) Delete (15, 1) : Just move up the smaller element

```
          (8, 3)
         /      \
    (13,8)      (20,4)
     /
  (4,9)
```

5. (14)

- Structure:

  Let the 3 keys be x, y, and z. A 3-dimensional range tree is a range tree on z, while each node in that tree is a range tree on y, and finally each node in that is *a sorted array* on x.

  The root contains all the records. A discriminator is chosen to be the median of the keys. According to the median key value, the records are split into two equal parts, one containing records no larger than the median goes to the left subtree, the other half goes to the right subtree. And the median is chosen and records are split recursively until the number of records is considerably small.

  Three parameters to discuss: P, S, and Q

- Preprocessing time P:

  Since we need the sorted array on x at every node, we need to sort the records on x first. The median can be found in linear time, and the splitting is done by scanning through the sorted array of the root, deciding which side a record should go on the fly. So after the splitting, the subarrays created remain sorted on x. All records are present at each level and there are at most logN levels. So the splitting on y takes NlogN at each level. Thus the total time to split on z, containing logN levels as well, takes N(logN)^2.

  Since it is larger than the sorting of x, NlogN, already, the complexity for the preprocessing is thus O(N(logN)^2).

- Extra space S:

  In the range tree on y, all records are present at each level. Thus each tree on y takes NlogN space. But there are total of logN levels of the range tree on z as well.

  So the total space required is O(N(logN)^2).

- Querying time Q:

  Given a query in the form (x1, x2, y1, y2, z1, z2), we first determine which nodes in the range tree of z are contained in the range [z1..z2]. Then at each node we perform another search on y to find out the nodes contained in [y1..y2]. Finally in each node of tree y we use binary search to determine where x1 and x2 are in the sorted array of x.

  There are logN nodes at tree z that are candidates, and inside each node there are logN nodes in tree y that may contain answers. Binary search on x takes O(lonN) time. So the total time is O((logN)^3). But we still need to report the answers, and the answer size might be larger than the search time.

  Thus, the final complexity is O((logN)^3 + |ans|)