Question 1 (12 points)

Consider the following splay tree:

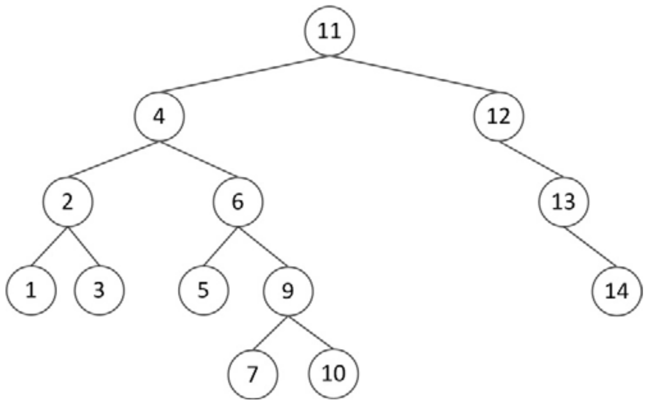
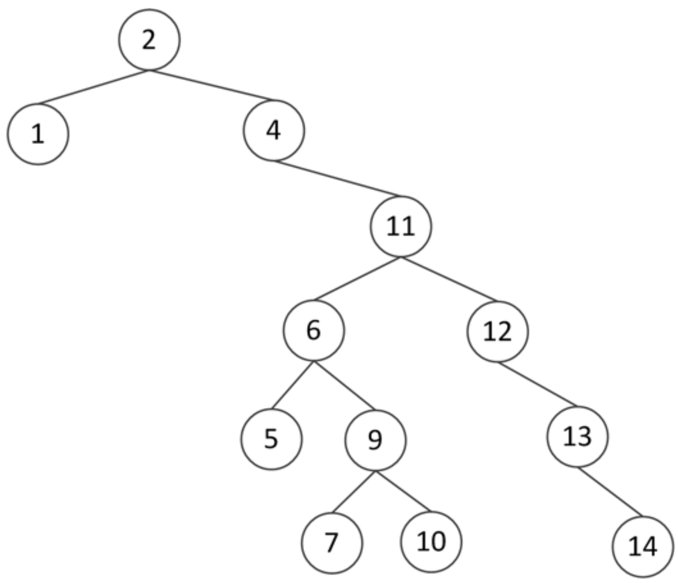


Figure 1

1) Perform a *delete* for the key 3 under the assumption that this is a *bottom-up* splay tree. Show each step. (6 points)

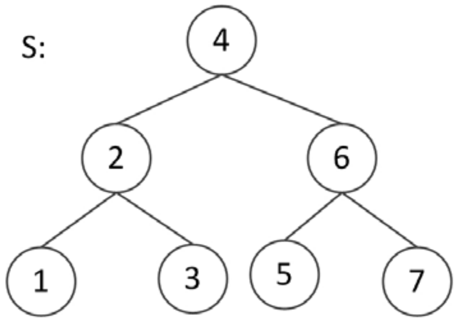
   Answer:



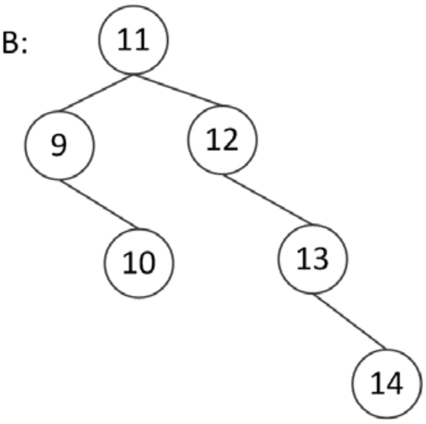
2) Perform a *split* from the tree of Figure 1 (not the resulting tree of part 1)) for the key 8 under the assumption that this is a *top-down* splay tree. Show each step. (6 points)

Answer:

S:



B:

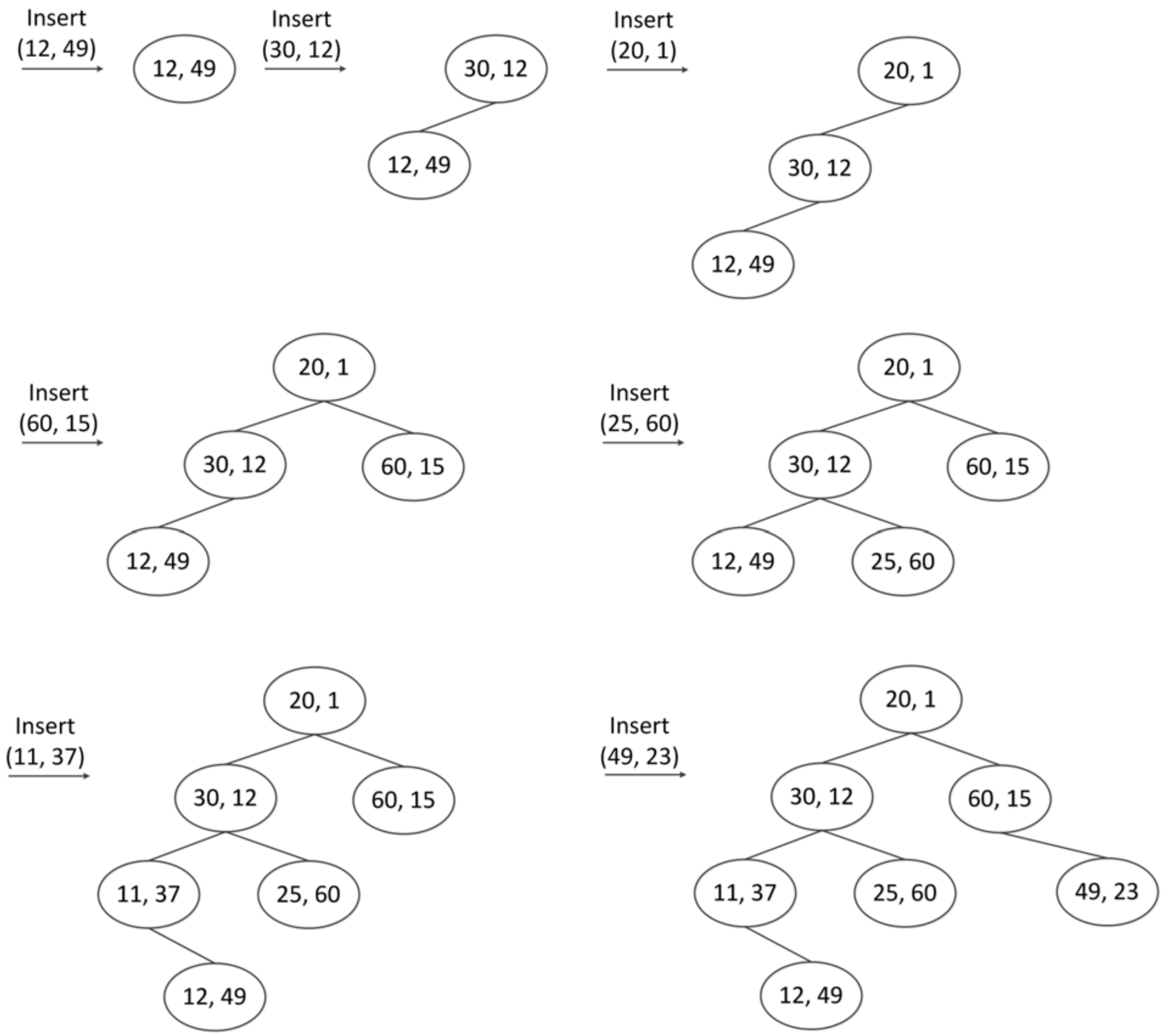


Question 2 (14 points)

A min radix priority search tree (RPST) can be defined as a set of pairs (x, y) over [0 … 63] of integers. Construct a min RPST by inserting the following pairs in the given order. Show the min RPST after each insertion.

(12, 49), (30, 12), (20, 1), (60, 15), (25, 60), (11, 37), (49, 23)

Answer:



Question 3 (10 points)

You are given a Bloom filter that consists of m = 11 memory bits and two hash functions f1() and f2() defined as below:

f1(k) = (3*k) mod m

f2(k) = (2*k) mod m

where k is a given key. Assume that all m bits of the Bloom filter are initially set to 0.

1) Show the Bloom filter bits following the insertion of the key 7, 12, 9. Show result after each insertion. (6 points)

Answer:

Insert key 7, f1(7) = 10, f2(7) = 3.

0	1	2	3	4	5	6	7	8	9	10
0	0	0	1	0	0	0	0	0	0	1

Insert key 12, f1(12) = 3, f2(12) = 2.

0	1	2	3	4	5	6	7	8	9	10
0	0	1	1	0	0	0	0	0	0	1

Insert key 9, f1(9) = 5, f2(9) = 7.

0	1	2	3	4	5	6	7	8	9	10
0	0	1	1	0	1	0	1	0	0	1

2) How can the resulting filter return a "maybe" for a key that was not inserted? (4 points)

Answer:

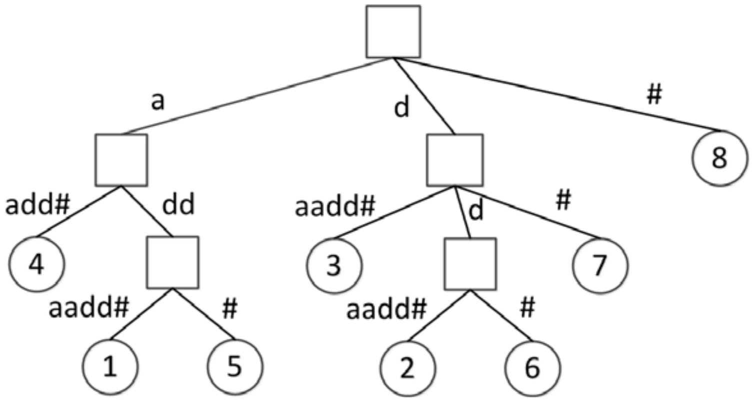
Let key = k, f1(k) = i_1, f2(k) = i_2. If the value at i_1 and i_2 are both already set to "1", a "maybe" is returned.

Question 4 (14 points)

1) Draw a clearly labeled suffix tree for the string *addaadd#*. (8 points)

Answer:

a	d	d	a	a	d	d	#
1	2	3	4	5	6	7	8



2) For R-Tree which is usually used as a spatial database index:

a) Briefly describe how to insert a node into R-Tree. (3 points)

Answer:

(*Leaf selection*) Follow the path from root to leaf, then insert node into subtree whose MBR is increasing least with the new inserted new rectangle.
(*Consider when to split*) If capacity exceeded, split set of M + 1 rectangles / MBRs into 2 sets A and B.

b) How does Quadratic Split Method work? Show it with an example. (3 points)

Answer:

Let S be the set of M + 1 rectangles to be partitioned.
(1) Find (a,b) in S so that area (area(MBR(a,b)) - area(a) - area(b)) is maximized.
(2) Assign remaining unassigned rectangles. Find an unassigned rectangle C to maximize:

|area(MBR(A,c)) – area(MBR(A)) - (area(MBR(B,c)) – area(MBR(B)))|

(3) Assign c to partition whose area increases least.
(4) Continue assigning in this way until all remaining rectangles must necessarily be assigned to one of the two partitions for that partition to have m rectangles.