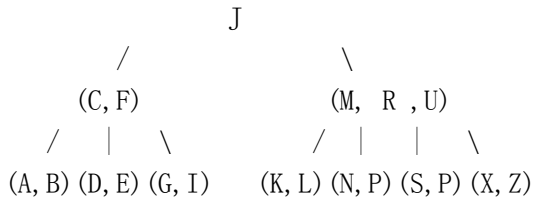
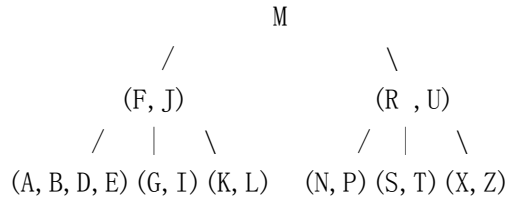
1. For B-trees,
(a) Delete the key 'C' from following B-tree of order 5. Draw the result.



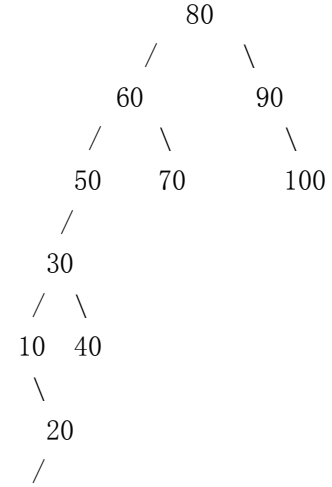
(solution)



(b) Suppose that n keys are inserted into an empty B-tree of order m. What is the maximum height h in the final B-tree?

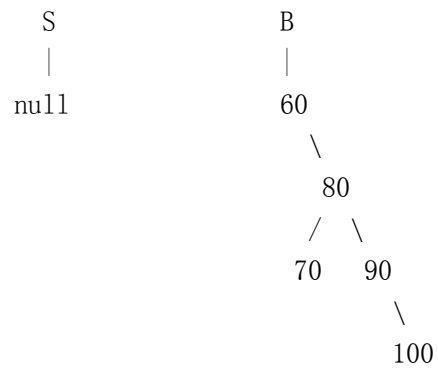
(solution)
1 + log_ceil(m/2) (n+1)/2

2. For splay tree,
(a) Delete for the 30 from the following tree. Asssume that this is a bottom-up splay tree.

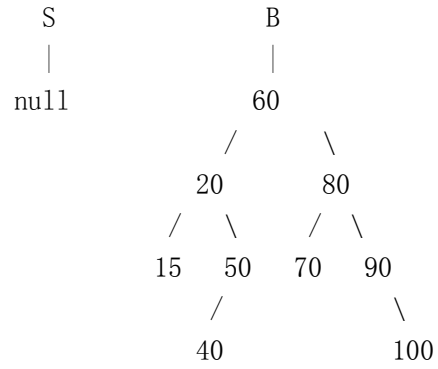


15

(solution)



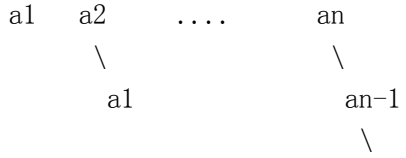
Step 2)

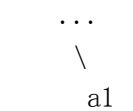


(b) What is the maximum height of a splay tree that is created as the result of n insertions made into an initially empty splay tree? Give an example of a sequence of inserts that results in a splay tree of this height.

(solution)
maximum height = n

Assume a1 > a2 > ... > an

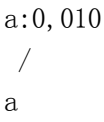




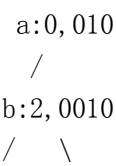
3. For Patricia,
(a) Insert the following keys into an initially empty instance of Patricia:
0101, 0010, 0011, 1011, 1000, 1101
Draw the patricia instance following each insertion.

(solution)
assume a node looks like: <label>:<bitnumber>,<key>

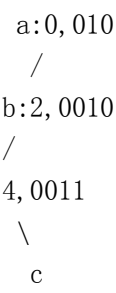
insert 0101



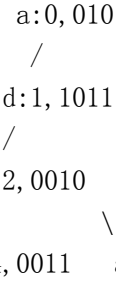
insert 0010



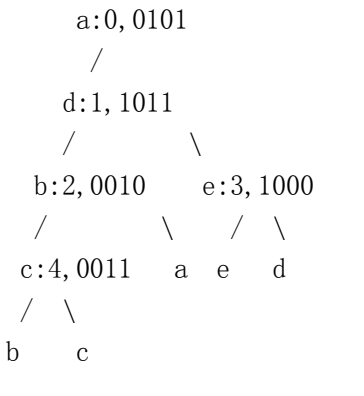
insert 0011



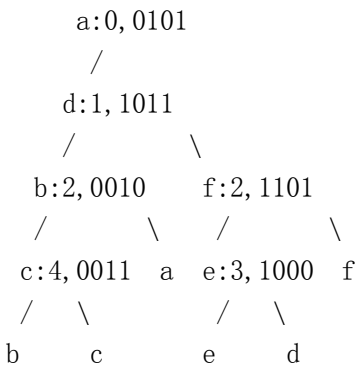
insert 1011



insert 1000



insert 1101

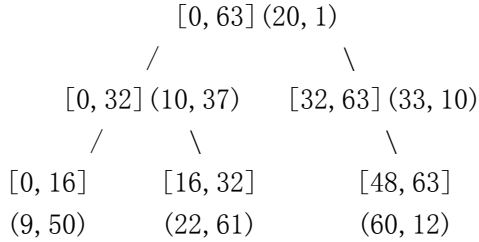


(b) Then, delete the key 1011 from the result of (a), draw the resulting instance.

4. A radix priority search tree can be defined as a set of ordered pairs [x,y] over 0..63 of integers maintaining a min-tree on y and a binary search tree on x.

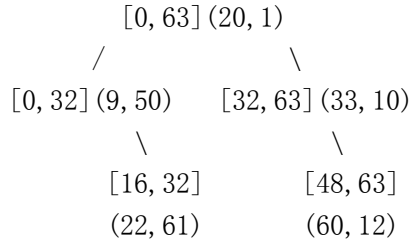
(a) Draw a radix priority search tree which contains pairs of [9,50], [33,10], [20,1], [60,12], [22,61] and [10,37].

(solution)



(b)

(solution)



5. Given is a k-d tree with the following information in each node: left[u] is the left child of u, right[u] is the right child of u, x[u] is the x-coordinate value of a point, y[u] is the y-coordinate value of a point, d[u] is the direction of the cut through(x[u],y[u])(horizontal or vertical). The root of the tree T is root[T].

(solution)

```
visit(u)
  if x_low<=x[u]<=x_high and y_low<=y[u]<=y_high then output(u)
  if d[u]=vertical
  then if x_low<=x[u] then visit(left[u])
       if x[u]<=x_high then visit(right[u])
  else if y_low<=y[u] then visit(left[u])
       if y[u]<=y_high then visit(right[u])
```

Above procedure is called with visit(root[T])