Assignment 3 Pengshengnan(Paul) Cheng V00838497

Question 1

Input : list of bolts and list of nuts

Ouput: list of matched bolts and nuts

1. If number of bolt = 1 then

   end
2. Randomly choose one bolt P.
3. Split the nuts into(group1,group2,group3). Put the nuts that are smaller than P into group1; put the nuts that fit P into group2; and put the nuts that are larger than P into group3.
4. Randomly choose one nut M from group2.
5. Split the bolts into (group4,group5,group6). Put the bolts that are smaller than M into group4; put the bolts that fit the M into group5; and put the bolts that are larger than M into group6.
6. Match group2 and group5
7. Recursively do steps 1-6 to match group1 bolts with group1 nuts, and group6 bolts with group3 nuts.

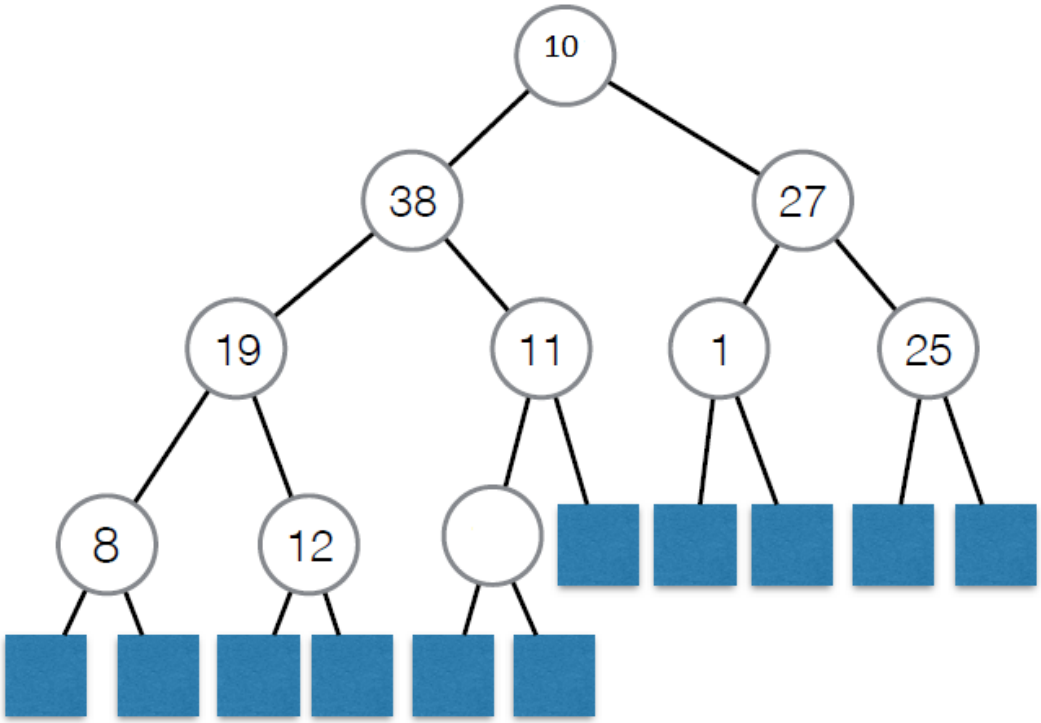
   End.

   It’s like quickselect, the average running time is O(nlogn)

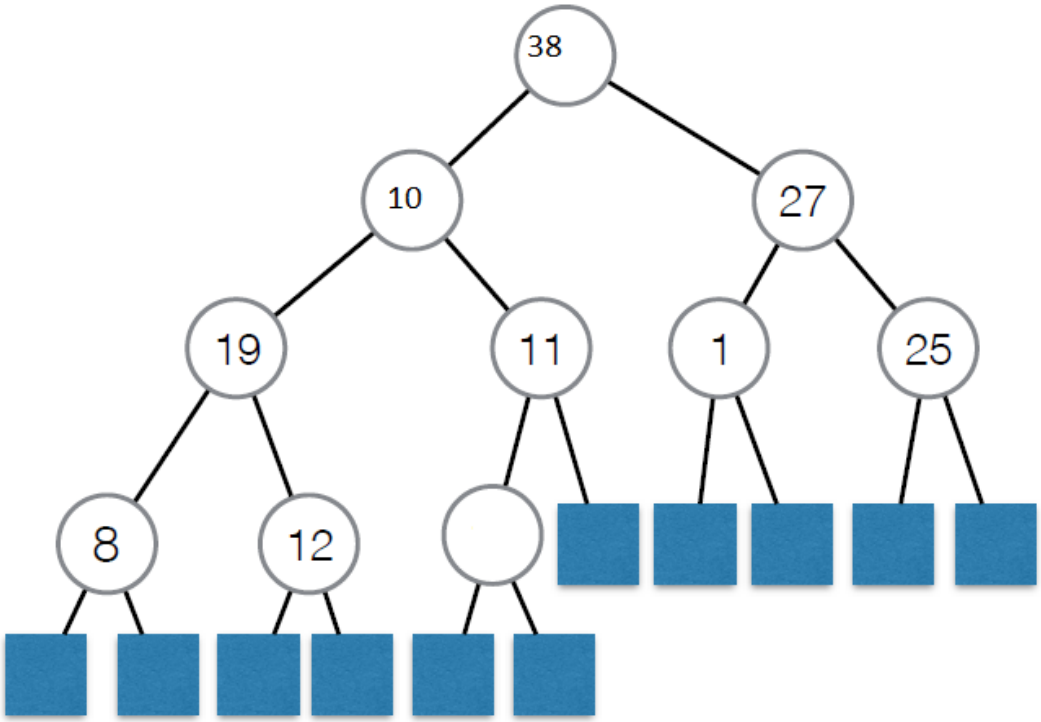
Question 2

(a) Find the right most element in the left sub tree and move to the top. And bubble out the maximum element from the tree

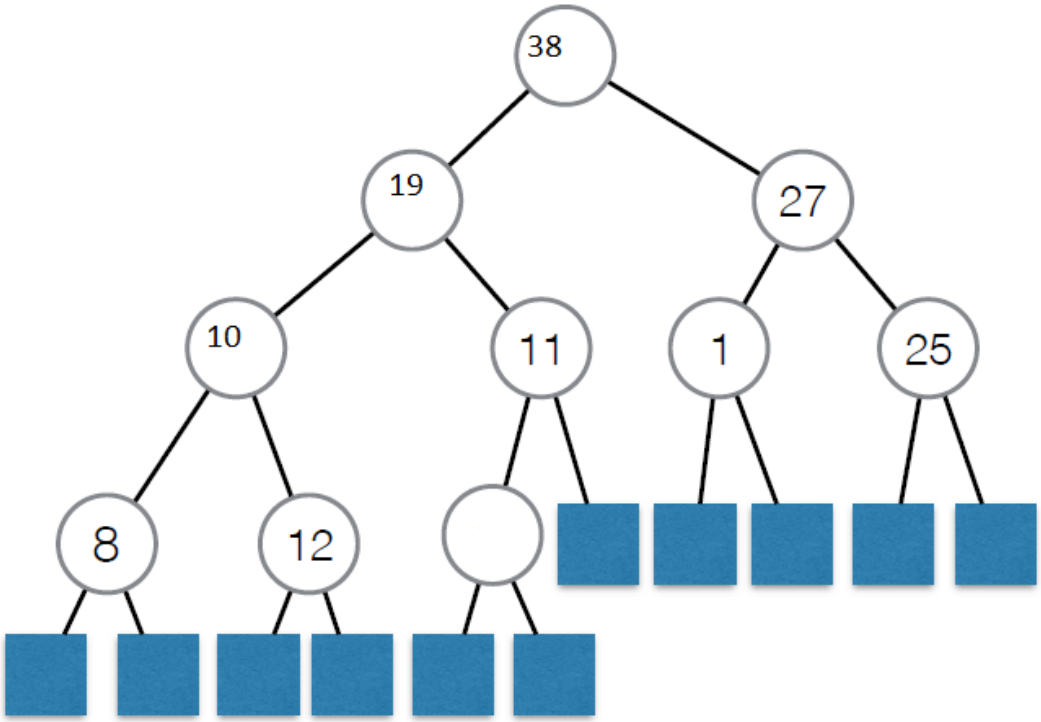
1.



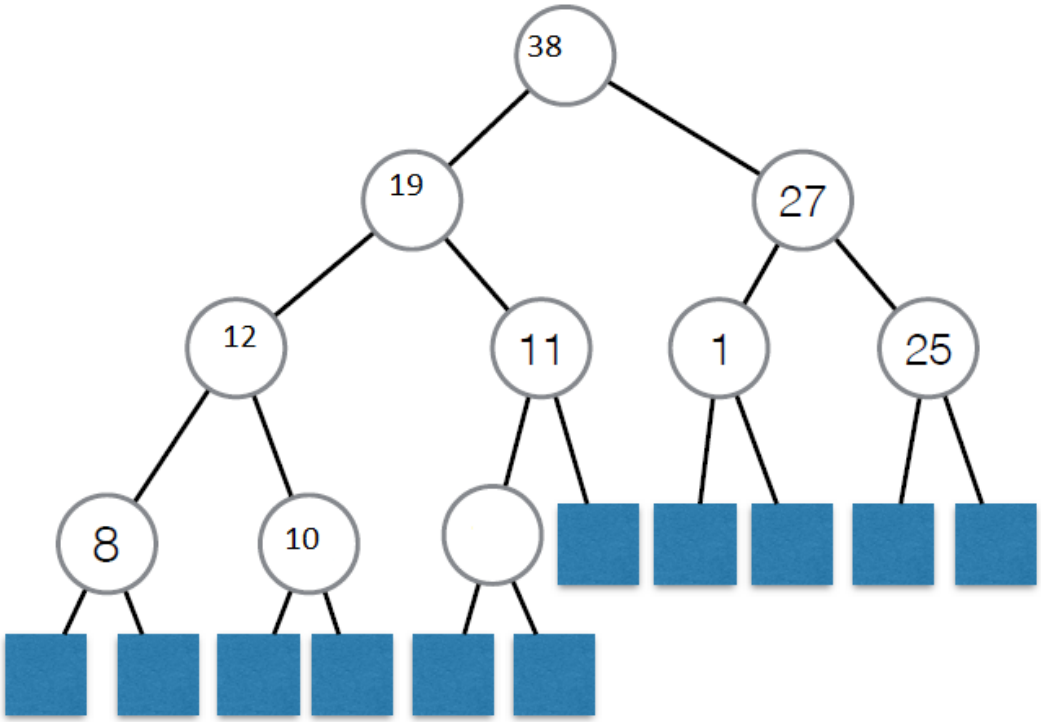
2.



3.

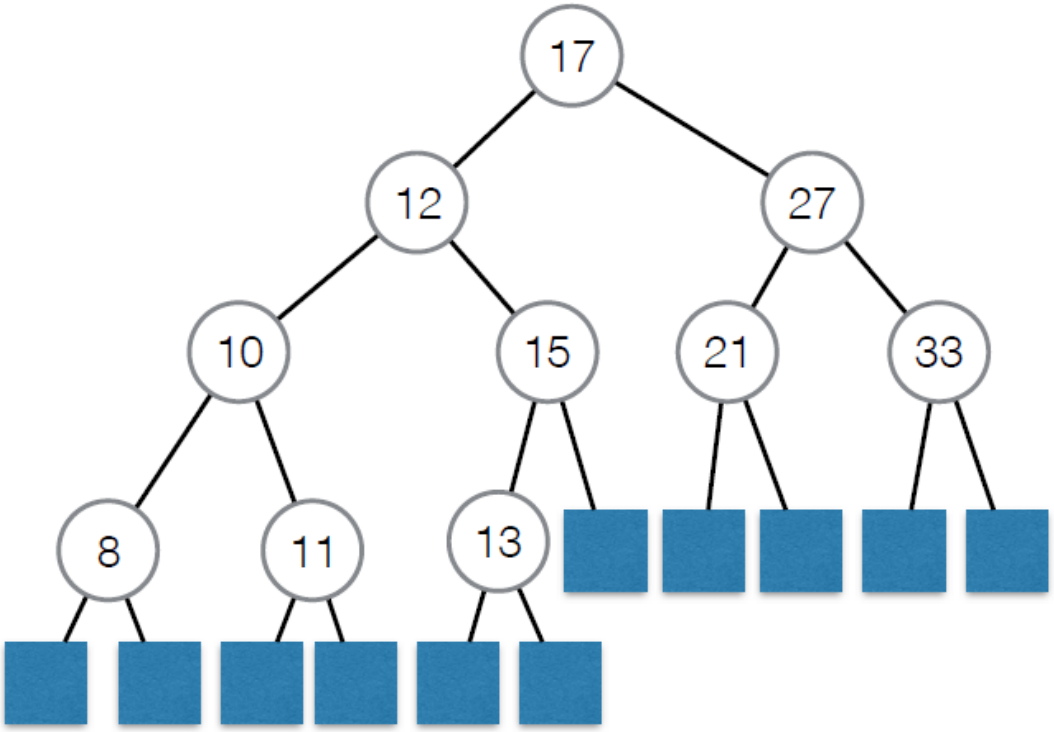


4.



(b)

1. Find the left most elements in the right sub tree and done.



Question3

1） When insert a node into a BST, it will compare the root all the way to the last leaf. Then insert the node. Treap has the same property, which is the left child key < root key; right child key > root key. Also treap has 2 children only. So treap is also a binary search tree with priority.

2)

TREAP-INSERT (T, x) inserts x into the treap T. It requires that x has defined key and priority values. We can use binary_search_tree_insert, rotate_right and rotate_left.

Input: treap,element

Output: treap

```
TREAP-INSERT (T, x)
    1 binary_search_tree_insert (T, x)
    2 while x = root[T] and priority[x] < priority[p[x]]
    3     if x = left[p[x]]
    4         do rotate_right
    5         else do rotate_left
```

Input : 2 nodes

Output: rotate right the nodes

```
rotate_right(root, parent):
    assert root.left == parent || root.right == parent
    let pivot = parent.left
```

```
    // R0
    parent.left = pivot.right
    // R1
    if parent == root.left:
        root.left = pivot
    else:
        root.right = pivot
    pivot.right = parent
input: 2 nodes
output: rotate left 2 nodes
rotate_left(root, parent):
    assert root.left == parent or root.right == parent
    let pivot = parent.right
    // L0
    parent.right = pivot.left
    // L1
    if parent == root.left:
        root.left = pivot
    else:
        root.right = pivot
    pivot.left = parent
```

3)

The worst-case is that, the treap only has a left subtree or right subtree. In this case, the height is n.

Question4

Input: integer n

Output: if the number is Halloween number or not

```
Halloween_number (int n):
  int m ←0
  int digit ←0
  table ← empty Hashtable
  while(n is not 1 and table does not contain n) do
     m ←0
     while (n > 0)
          digit = n%10
          m += digit * digit
          n = n/10
     end while
     n = m
  end while
  return n ==1
```