**BY: Fadi Alahmad Alomar 120180049**

Trees

# Question 7

Number of nodes = N

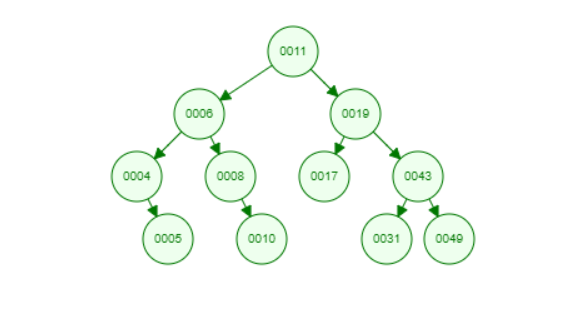
Number of nodes at level k = 2k

Number of levels = h

2h+1 = N

h = log2(N) - 1 = O(log(N))

# Question 8



# Question 10  
FUNCTION searchNode(Tree,value):  
 while Tree != NULL and Tree.value != value:  
 if value > Tree.Value:  
 Tree <- Tree.right  
 else:  
 Tree <- Tree.left  
 if root == NULL:  
 return False  
 return True  
  
# Question 11  
FUNCTION searchWithParentsReturn(root,value):  
 rootParent <- root  
 while root != NULL and root.value != value:  
 rootParent <- root  
 if value < root.value:  
 root <- root.right  
 else:  
 root <- root.left  
 if root == NULL:  
 return False ,None, rootParent  
 return True , root,rootParent  
  
FUNCTION getNextBigger(root):  
 root <- root.right  
 rootOld <- root  
 while root != NULL:  
 rootOld <- root  
 root <- root.left  
 return rootOld  
  
FUNCTION delNode(root,node):  
 exist <- searchWithParentsReturn(root,node)  
 if not exist[0]:  
 return NULL  
 else:  
 if root.left == NULL and root.right == NULL:  
 if root == exist[2].right:  
 exist[2].right = NULL  
 else:  
 exist[2].left = NULL  
 return NULL  
 else:  
 n = getNextBigger(root)  
 if exist[2].right == root:  
 exist[2].right = n  
 else:  
 exist[2].left = n  
 delNode(root.right,n.value)  
  
# Question 13  
FUNCTION leavesNum(root):  
 if root == NULL:  
 return 0  
 if root.left == NULL and root.right == NULL:  
 return 1  
 return leavesNum(root.left)+leavesNum(root.right)  
  
# Question 14  
FUNCTION inOrderSort(root,x,n=0)  
 if root == NULL:  
 return NULL  
 inOrderSort(root.left,x,n)  
 x[n] = root.value  
 inOrderSort(root.left,x,n+1)  
 return x  
FUNCTION sort(arr):  
 myTree <- BSTree()  
 for i=0,i< arr.len() ,i++:  
 myTree.addNode(arr[i])  
 x <- array[arr.len()]  
 arrSorted <- inOrderSort(root,x)  
  
# Question 15  
def maxPathSum(root):  
 if root is None:  
 return 0  
 if root.left is None and root.right is None:  
 return root.value  
 s = maxPathSum(root.left), maxPathSum(root.right)  
 return max(s) + root.value  
  
# Question 16  
 def mirror(root):  
 if root is None:  
 return  
 root.left, root.right = root.right, root.left  
 mirror(root.left)  
 mirror(root.right)

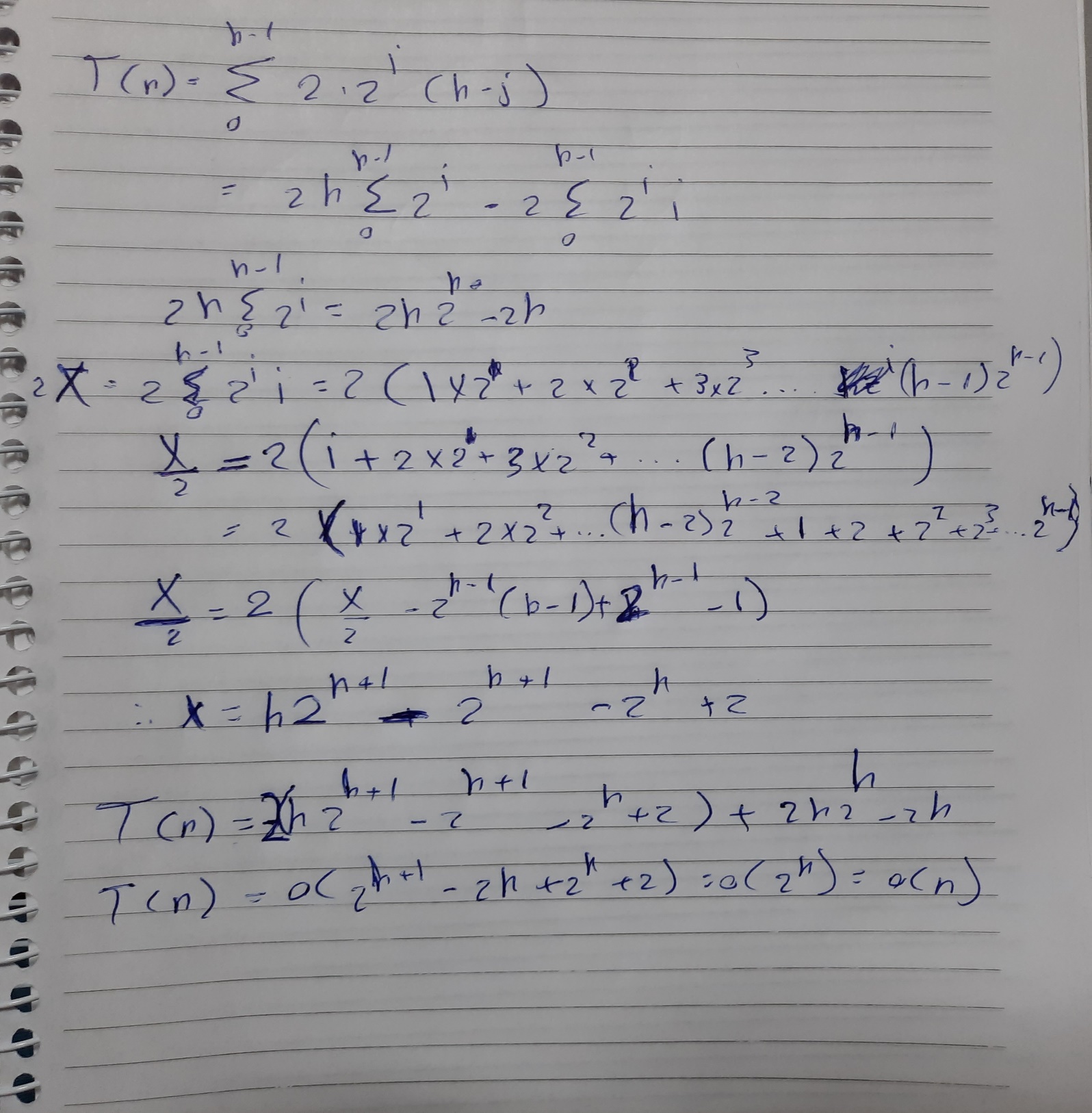
Heaps

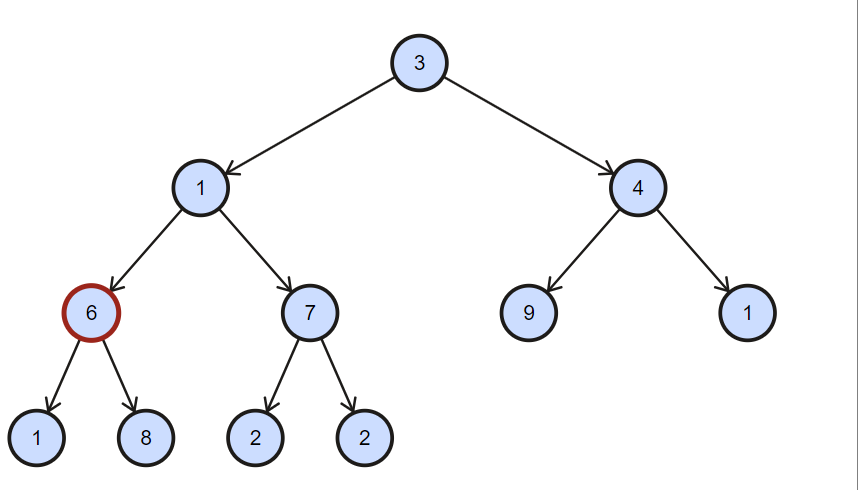
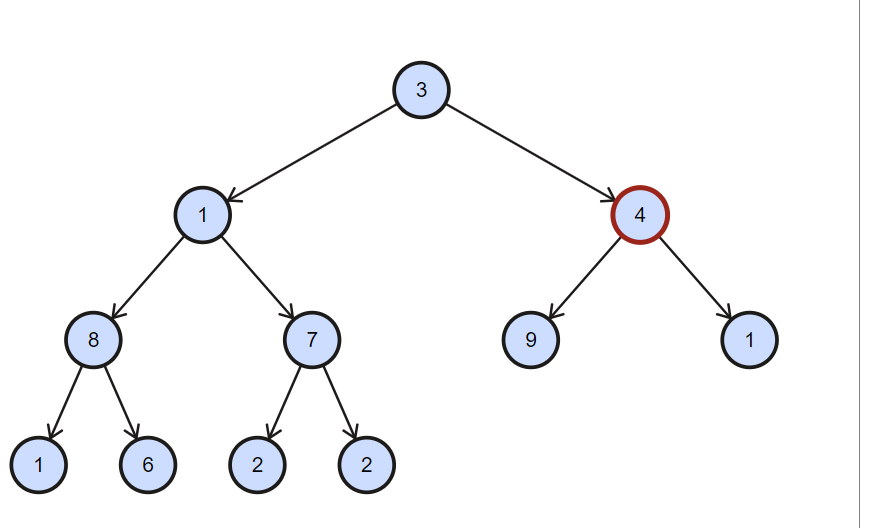
**# Question 1:**

A maximum heap is a binary tree where the value of the parent is bigger than both of his children, thus having the maximum value of the tree at top.

A minimum heap is a binary tree where the value of the parent is smaller than both of his children, thus having the minimum value of the tree at top.

It is possible to have minheap and BST is we allow for duplicates.

**# Question 2:**

# Question 3:

