Ch.5.1-5.6 Trees 參考答案

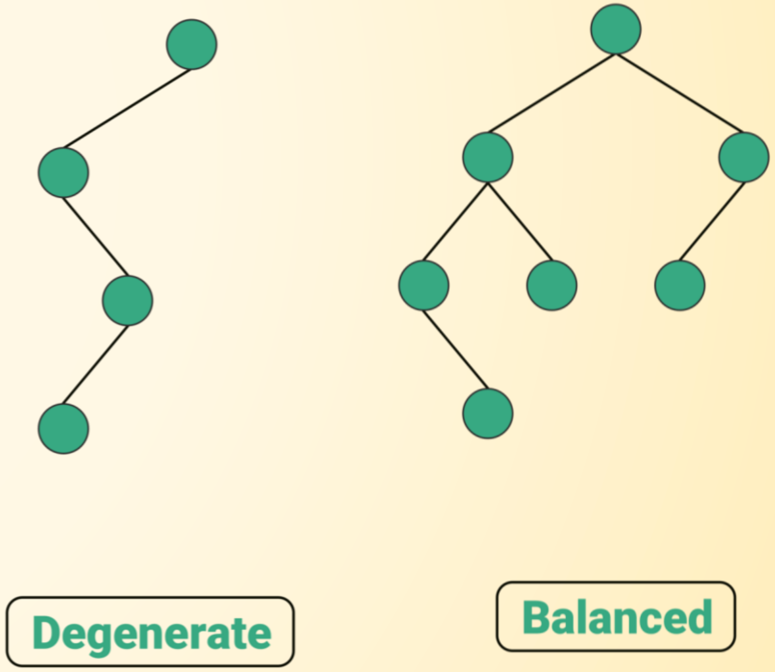
Q1

Please give an example to each of the following binary tree types.

- Full binary tree
- Complete binary tree
- Skewed binary tree
- Balanced Binary Tree
- Degenerate (or pathological) tree

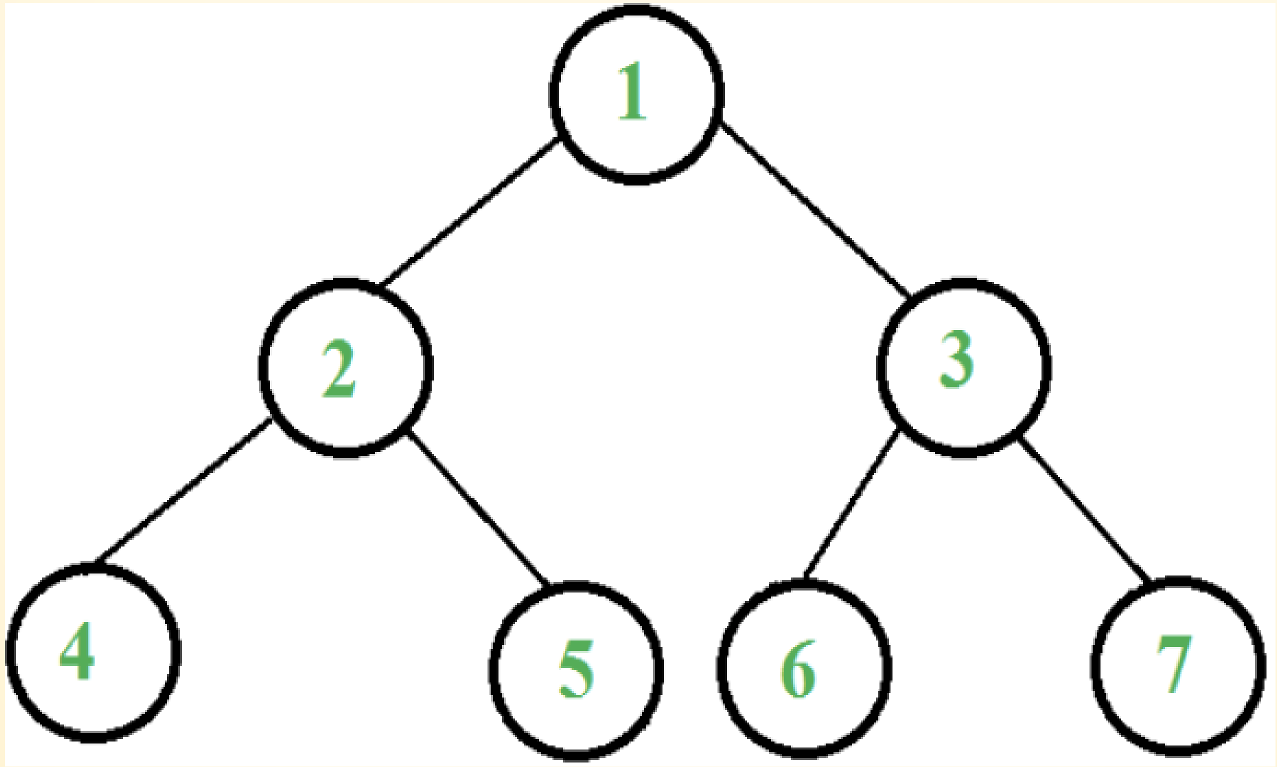
A1:

- **Balanced binary tree : a binary tree in which the left and right subtrees of every node differ in height by no more than 1.**
- **Degenerate binary tree : a binary tree in which each node has 1 child.**



Q2:

Design a non-recursive method to traverse a binary tree in postorder.



A2:

參考: https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/

Algorithm 1: *postorderIterative(R)*

```
Input: root
stack S;
node currentNode;
if (R == Null) then
    return;
end
S.push(R);
while S is not empty do
    currentNode ← S.pop();
    if (!currentNode → right)&&(!currentNode → left) then
        cout << currentNode;
    else
        S.push(currentNode);
    end
    if (currentNode → right) then
        S.push(currentNode → right);
        currentNode → right = Null;
    end
    if (currentNode → left) then
        S.push(currentNode → left);
        currentNode → left = Null;
    end
end
```

Q3:

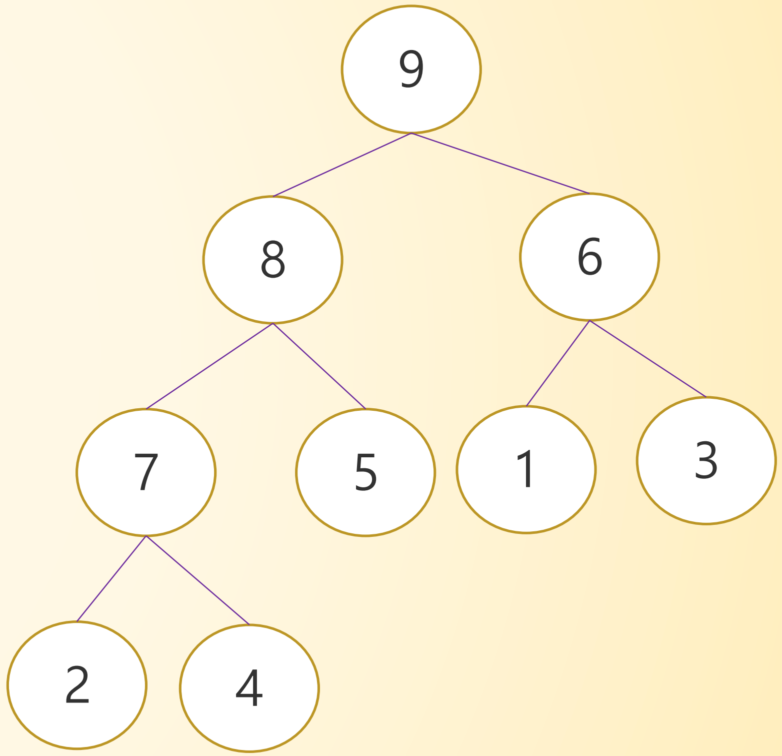
Please construct a max-heap in both **bottom-up** and **top-down** method from the following array A, and explain how you do it and time complexity.

A[9] = { 2, 5, 1, 8, 9, 3, 6, 4, 7 }

Try to implement your Max Heap as well as <u>its related operations</u> using C++.

A3: [Top-down]

- Insert item $n\ times$
  - ➔ worst case: n *O(log n) = O(n log n)
  - ➔avg. case: n *O(1) = O(n) [supplement]



Insert - Avg. case : O(1)[supplement]

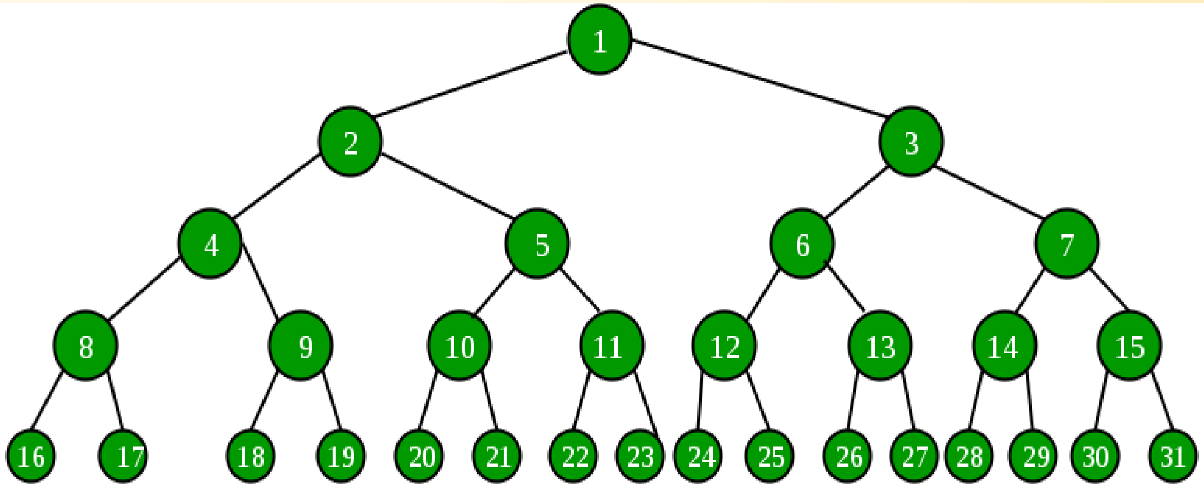
$n/2$ of the values are at height h (the leaves of the tree)

$n/4$ of the values are at height $h-1$

$n/8$ of the values are at height $h-2$

...

1 of the values are at height 0 (the root of the tree)



So, our new value has probability $1/2$ of being at height h, which requires one compare and no swaps. It has probability $1/4$ of being at height $h-1$, which requires two compares and one swap, etc.

expected (average) work to insert $= \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{16} \cdot 4 + \dots$

The infinite series $\sum \frac{k}{2^k}$ converges (to 2, in fact), so the average amount of work is $O(1)$.

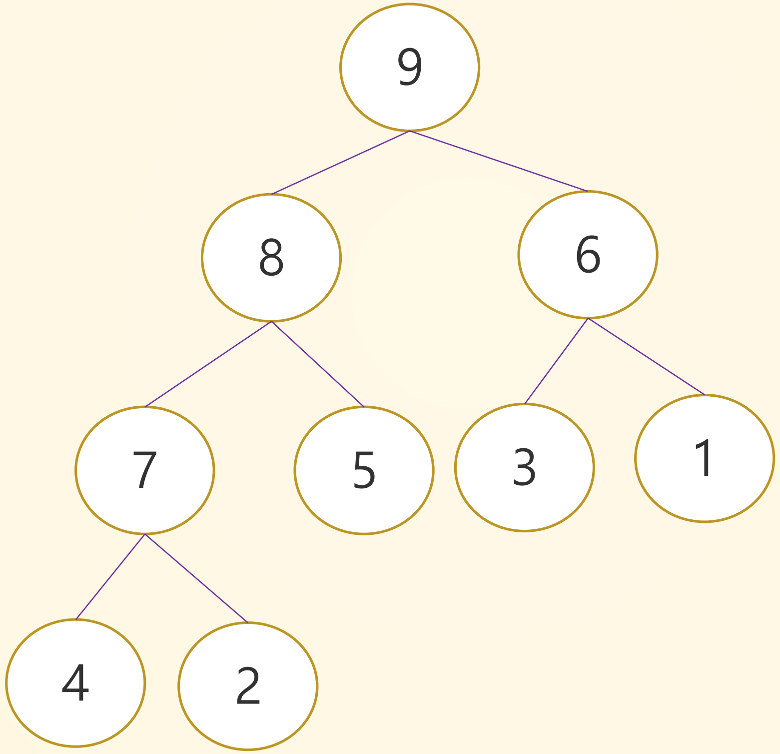
Insert - worst case : O(log n) [supplement]

Note: Can we optimize heap insertion in the worst case?

(hint: from O(log n) to O(log log n))

A3: [Bottom-up]

- O(n)

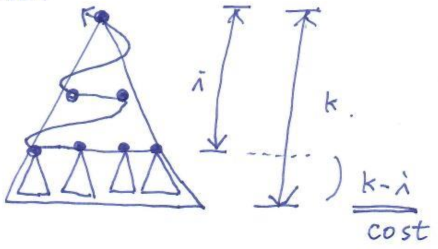


A3: [Bottom-up][supplement]

Bottom-up

O(n) step 1. Construct complete binary tree for given. input data.

O(n) $\boxed{\text{step 2}}$. From the last parent to the root node, heapify each subtree.



Time Complexity :

$$k = \lceil \log_2 (n+1) \rceil$$

the level of subtree's root $= i$

the number of level-i subtree $\leq 2^{i-1}$

$\Rightarrow$ total cost $= 2^{i-1} \cdot (k-i)$

$$\Rightarrow \sum_{i=1}^{k-1} 2^{i-1} \cdot (k-i) = S$$

$$S = 2^0 (k-1) + 2^1 (k-2) + \cdots + 2^{k-2} \cdot 1$$

$$2S = 2^1 (k-1) + 2^2 (k-2) + \cdots + 2^{k-1} \cdot 1$$

$$2S - S = -(k-1) + 2^1 + \cdots + 2^{k-1}$$

$$S = \left(\frac{2^k - 2^1}{2-1}\right) - (k-1) = 2^k - k - 1 \quad (k \doteqdot \log_2 n)$$

$$= n - \log_2 n - 1$$

$\Rightarrow O(n)$

Mix-Heap Implementation

參考: https://www.geeksforgeeks.org/binary-heap/

Q4:

Please re-construct all possible binary trees based on each following condition.

A. Inorder traversal sequence : 1 2 3
B. Preorder traversal sequence is equal to postorder traversal sequence
C. Postorder traversal sequence is equal to inorder traversal sequence

A4:

A.

```
1          1                    3           3
 \          \         2        /           /
  2          3       / \      1           2
   \        /       1   3      \         /
    3      2                    2       1
```

B. Empty and root only

C. Empty, root only and left skewed binary tree

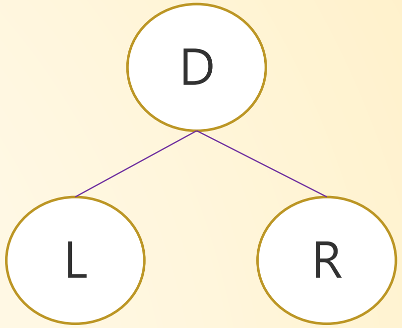
Q5:

Answer True or Fales to the following questions and explain your answers.

A. A binary tree re-constructed, based on a given postorder traversal sequence and an inorder traversal sequence, is unique.

B. A binary tree re-constructed, based on a given postorder traversal sequence and a preorder traversal sequence, is unique.

A5:



a. **True**

Postorder : LRD

Inorder : LDR

We can always find the root node (i.e. D) from postorder sequence, then find the left and right subtrees (i.e. L & R) from inorder sequence.

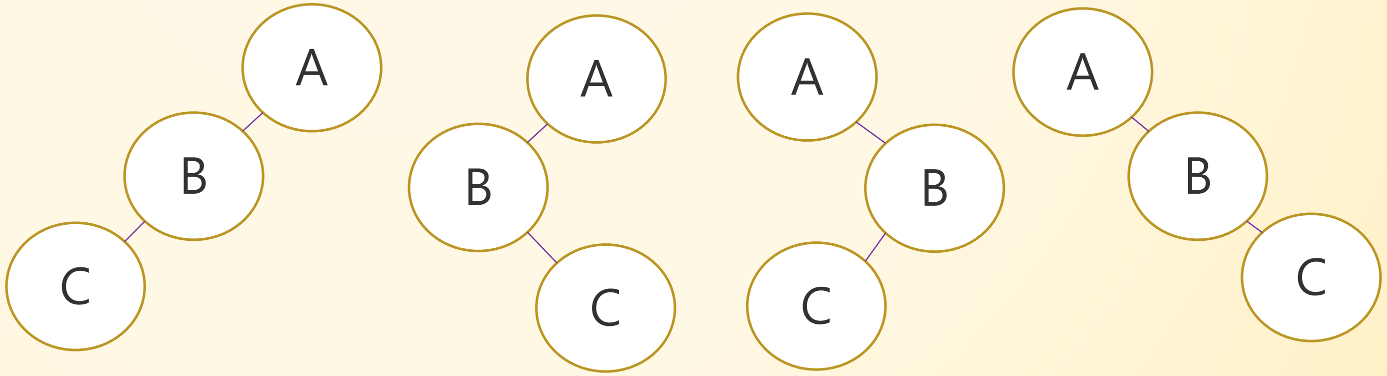
→ unique

b. **False**

E.g.

Preorder : ABC

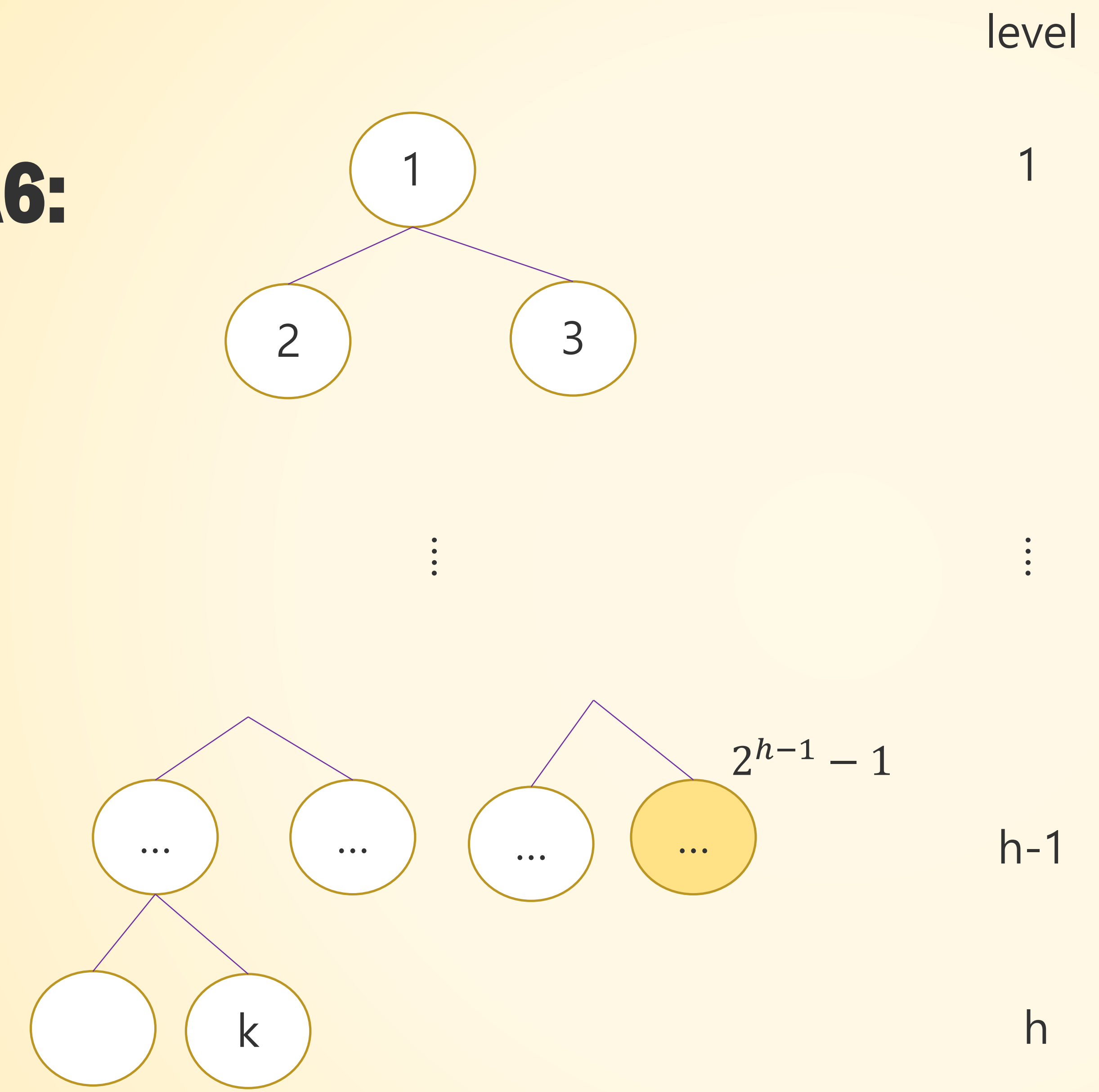
Postorder : CBA



Q6:

Prove that the node of index *k* in complete binary tree is at the height equals $\lfloor \log_2 k \rfloor + 1$ (The height of the root is 1.) $(k \geq 1)$

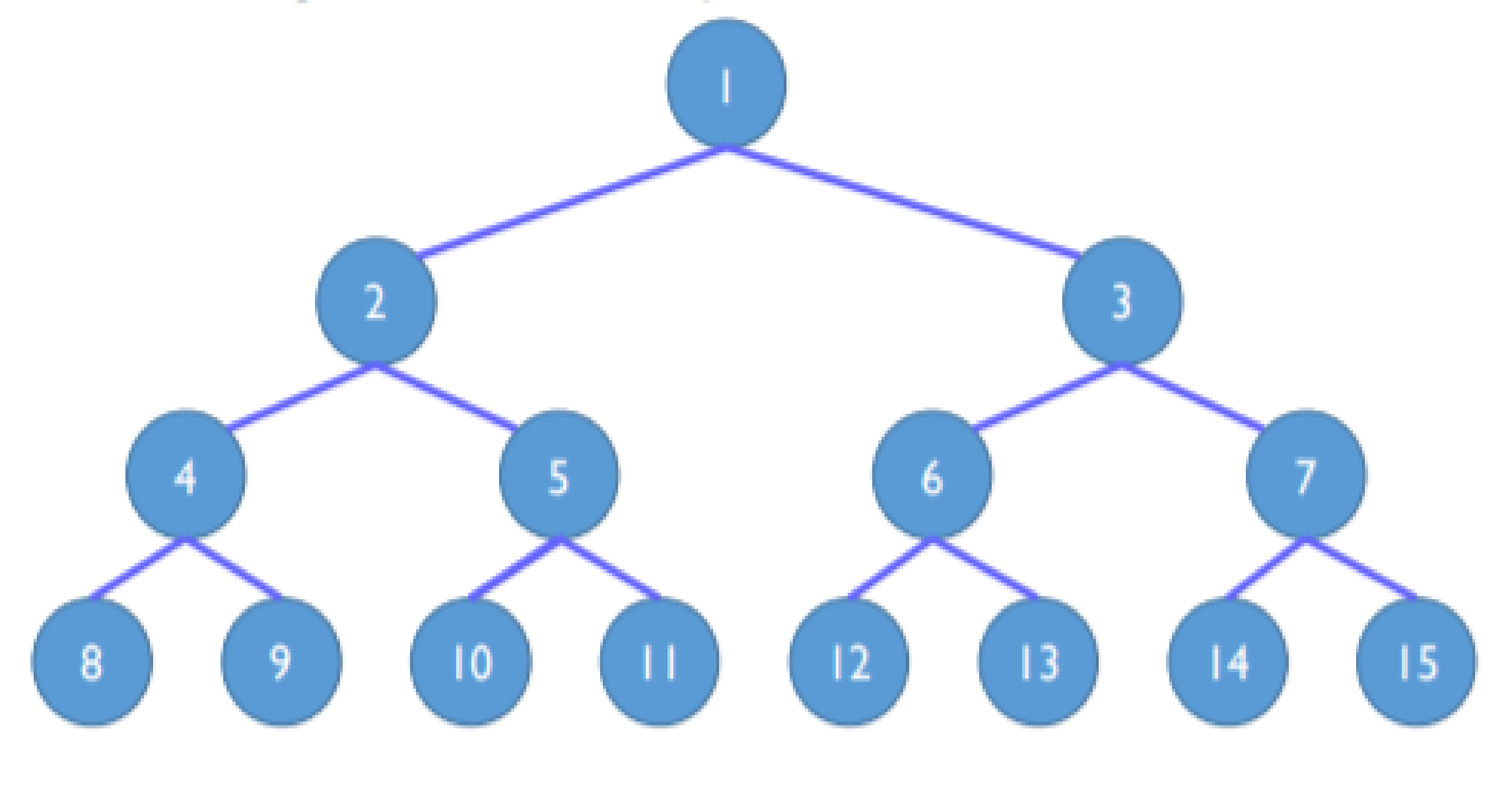
A6:



$$2^{h-1}-1<k$$
$$\Rightarrow 2^{h-1}\le k$$
$$\Rightarrow h-1\le \log k$$
$$\Rightarrow h=\log_2 k+1$$

Q7:

Prove that a full binary tree of depth *k* has 2^k-1 nodes.



A7:

$$2^0 + 2^1 + \cdots + 2^{k-1} = 2^k - 1$$

Q8:

Can you apply Max/Min heap for sorting algorithm? If so, explain how to do it.

A8:

Heap sort :

step 1 : construct max heap by bottom-up method → O(n)

step 2 : find max item and delete it → O(log k)

step 3 : go step 2 until the heap is empty → n times

Time complexity:

$$O(n) + O\left(\sum_{k=1}^{n} \log k\right) = O(\log(n!)) = O(n \log n)$$

The disadvantages of heap sort

What are the disadvantages of heap sort?

(Hint: spatial locality, unnecessary comparisons and exchange)

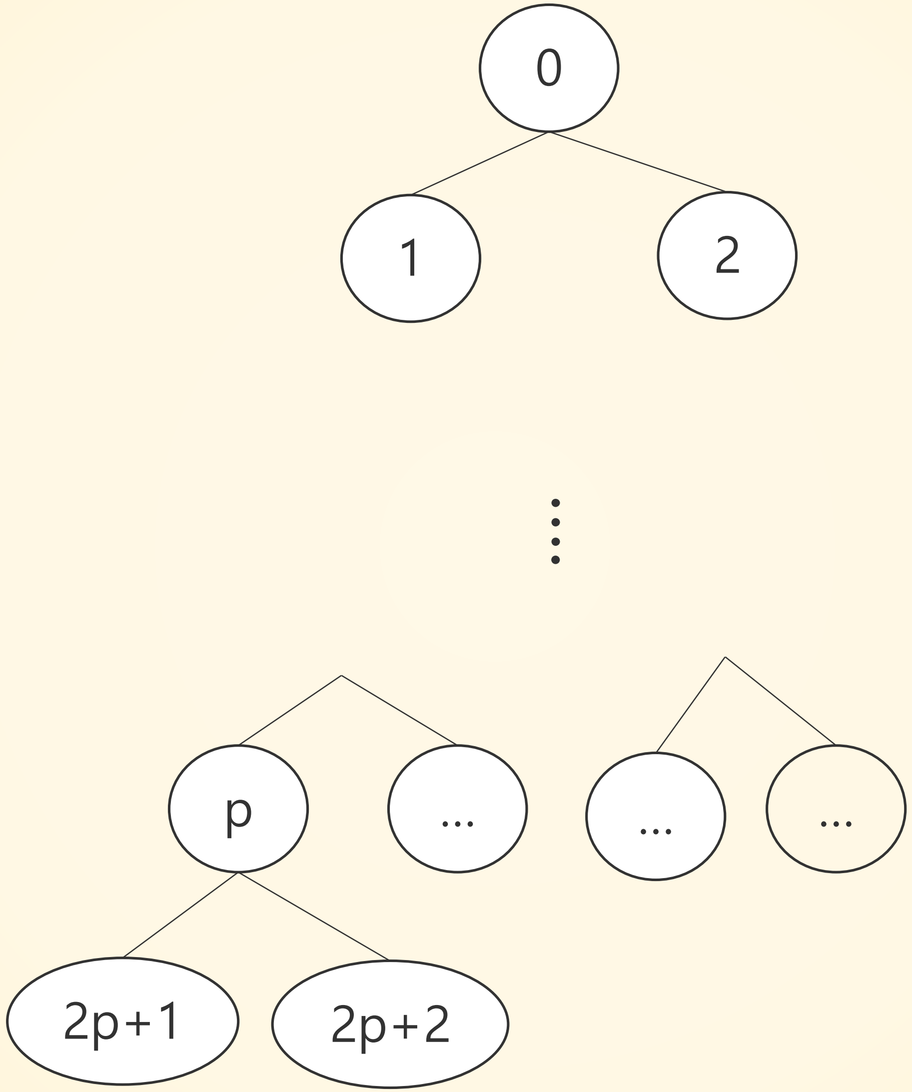
参考: https://www.cs.auckland.ac.nz/software/AlgAnim/qsort3.html

Q9:

If we use an array to represent a binary tree of n nodes, the indices of nodes can start from 0 to n-1. If the index of a father node is “p”, what are the indices of its left son and right son?

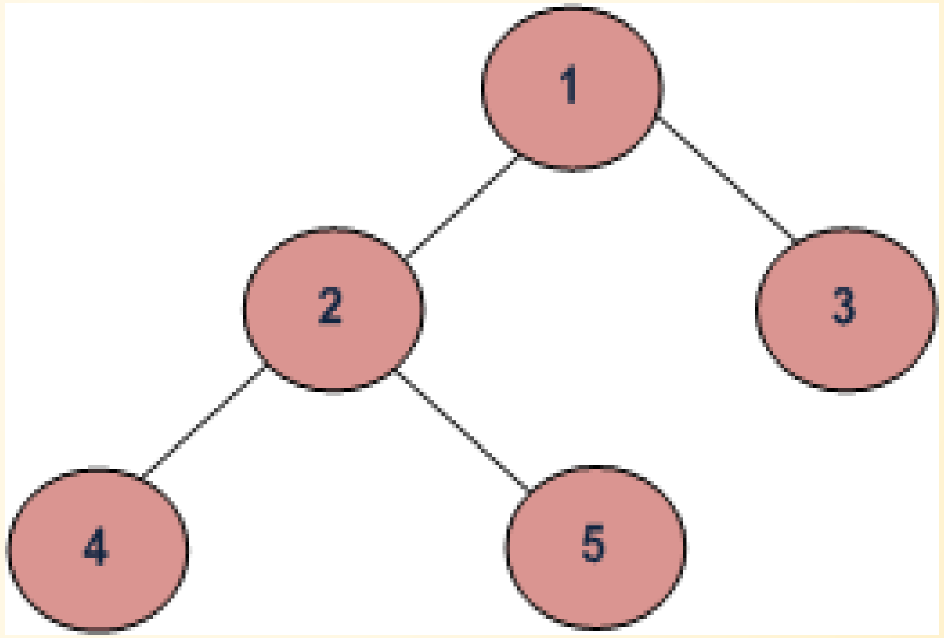
A9:

- 2p+1, 2p+2



Q10:

Please write a program to calculate the size of a binary tree. The size of a tree is the number of elements present in the tree.   For example, the size of the tree below is 5.



A10:

Algorithm 1: $NodeCount(T)$

Input: binary tree

if $(T = NULL)$ **then**

  return 0;

else

  $nL \leftarrow NodeCount(T \rightarrow LeftChild)$;

  $nR \leftarrow NodeCount(T \rightarrow RightChild)$;

  return $nL + nR + 1$;

end

Q11:

- Please write a program to count the number of leaf nodes in a binary tree. You must consider two cases: the binary tree is implemented by an array and by a linked list.

A11:

Algorithm 1: $LeafCount(T)$

```
Input: binary tree
if (T = NULL) then
    return 0;
else
    nL ← LeafCount(T → LeftChild);
    nR ← LeafCount(T → RightChild);
    if ( (nL + nR) > 0) then
        return nL + nR;
    else
        return 1;
    end
end
```