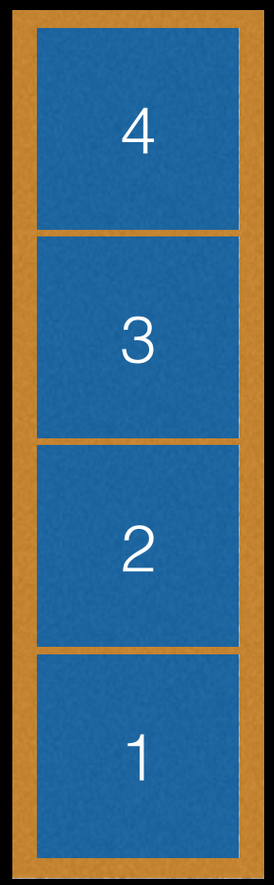
CS235101
Data Structure
期中考題講解

5. (12%) Stack

- Use a stack to evaluate the expressions. Please write down the type of the notation(prefix, infix, or postfix), the computed result, and the maximum number of elements stored in the stack at any time moment during the computation process.
- Ex: 1 2 3 4 * + + Answer: postfix, 15, 4
- (a) (6%) 4 2 5 * 3 - 6 * + 7 - 1 +
- (b) (6%) + - + 3 * - * 5 4 2 1 7 6

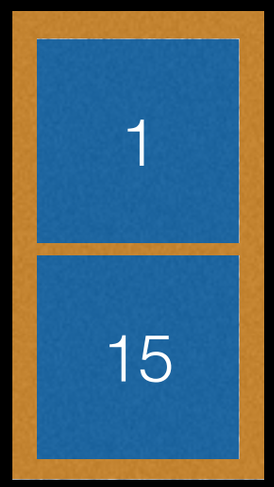
5. (12%) Stack

- Example: 1 2 3 4 * + +



5. (12%) Stack

- Example: + 1 +2 * 3 4



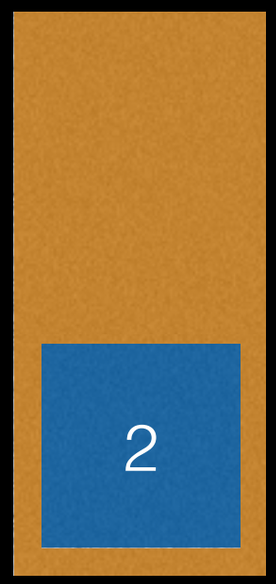
5. (12%) Stack

- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -

Preorder

5. (12%) Stack

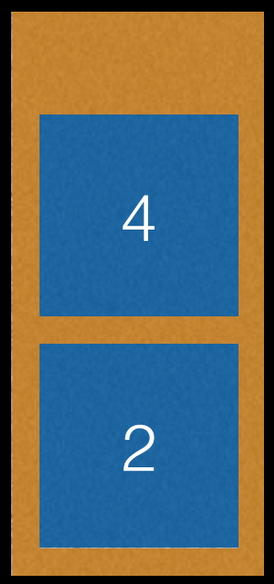
- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



Preorder

5. (12%) Stack

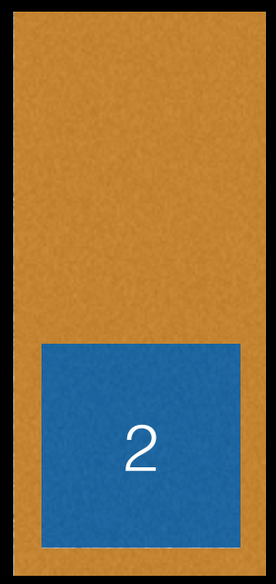
- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



Preorder

5. (12%) Stack

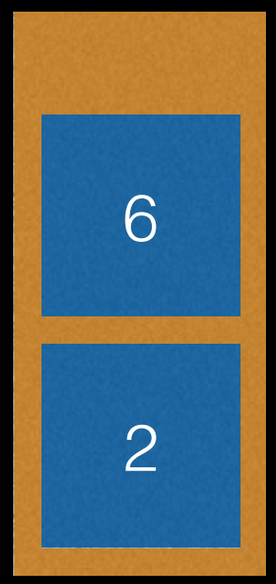
- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



Preorder

5. (12%) Stack

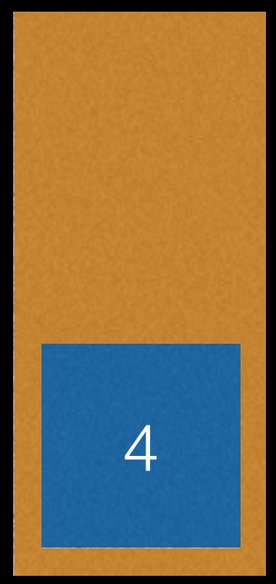
- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



Preorder

5. (12%) Stack

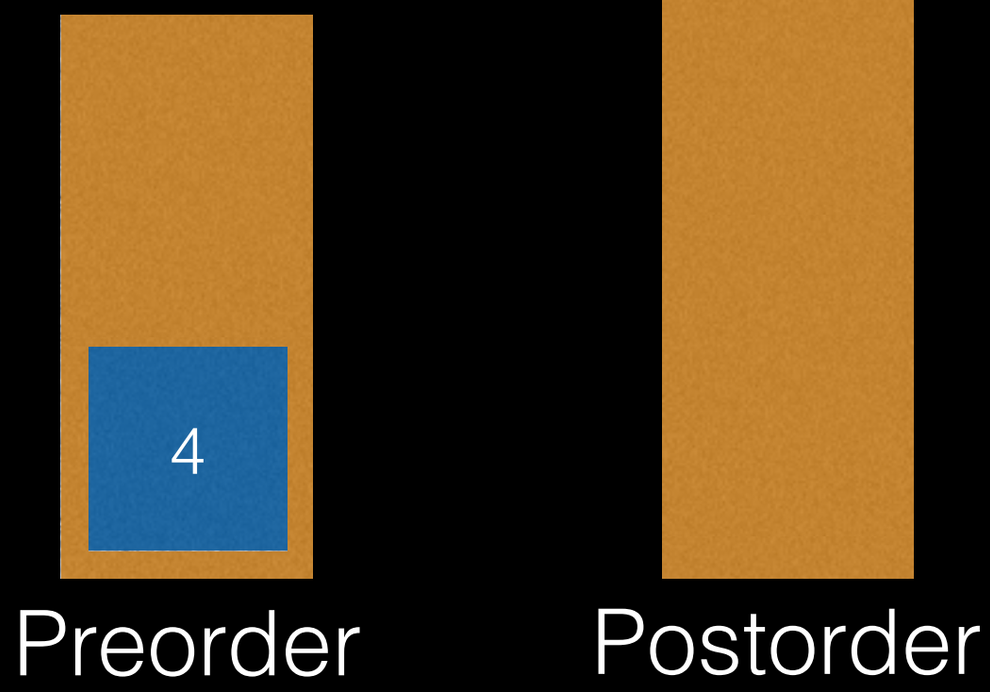
- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



Preorder

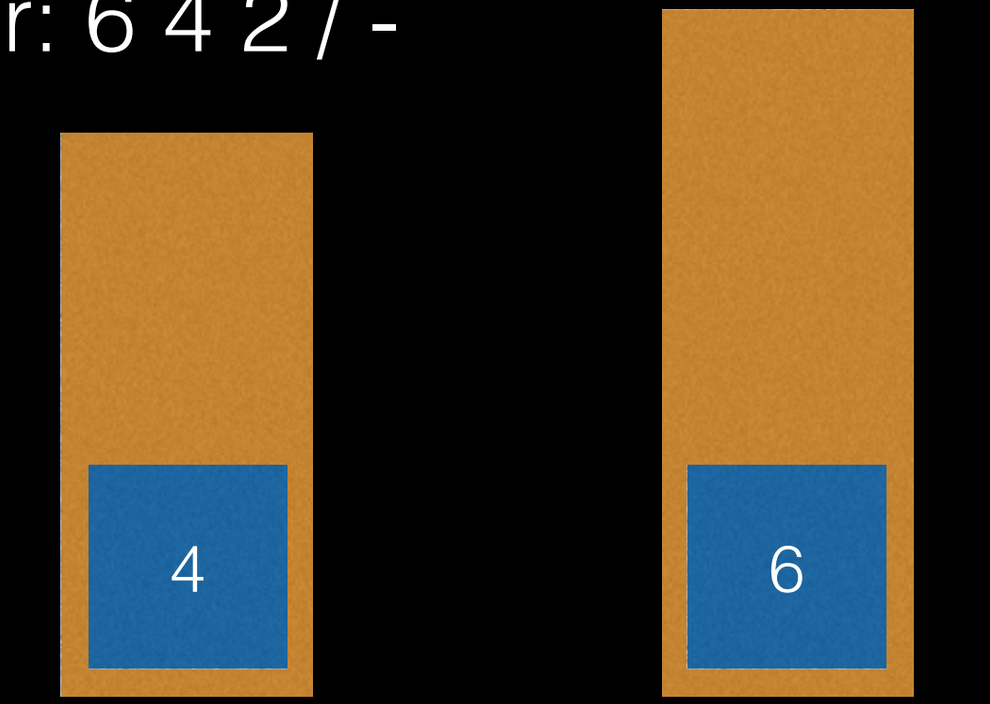
5. (12%) Stack

- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



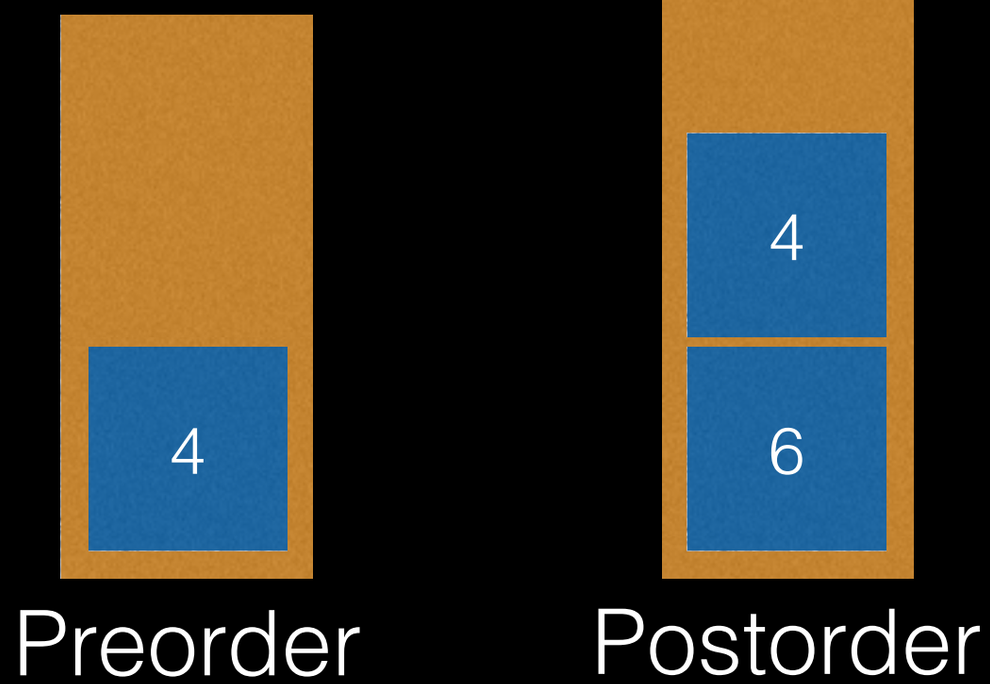
5. (12%) Stack

- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



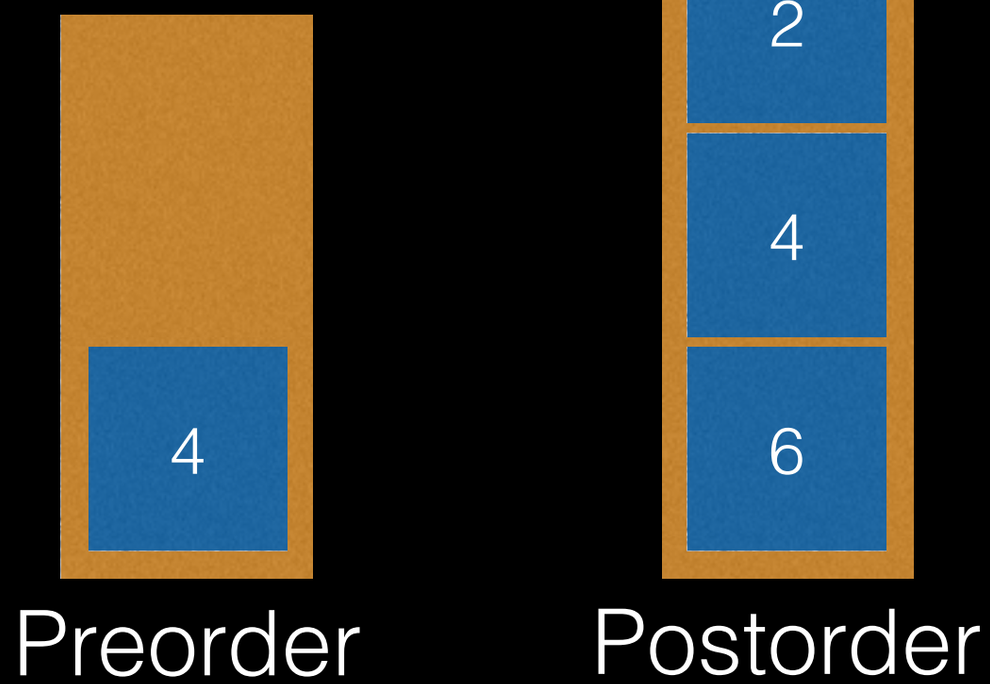
5. (12%) Stack

- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



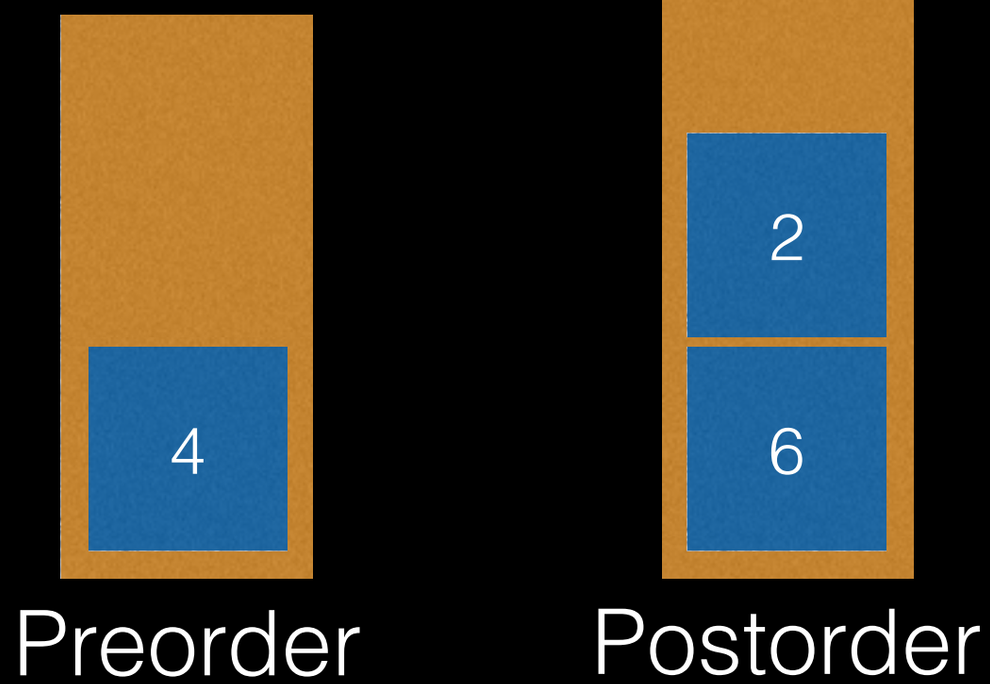
5. (12%) Stack

- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



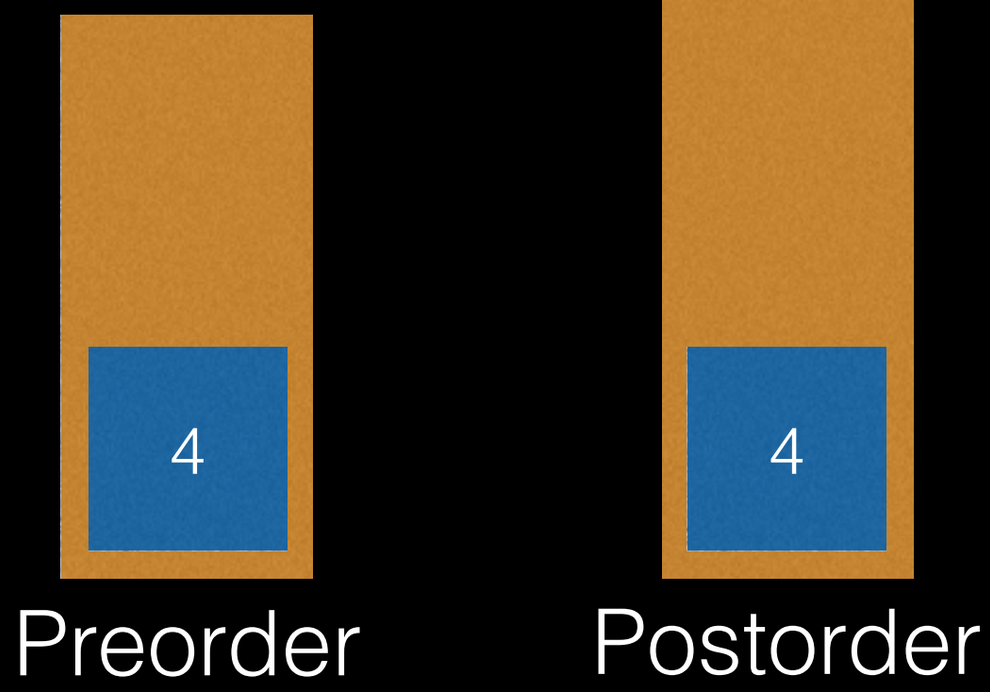
5. (12%) Stack

- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



5. (12%) Stack

- Example: 6-4/2
- preorder: - 6 / 4 2
- postorder: 6 4 2 / -



8. (8%) Tree

- Given the output of Preorder and Inorder traversal below, please write the output of Postorder.
- Preorder: ABCHDEFG
- Inorder: HCBDAFEG

8. (8%) Tree

- Preorder 

- Inorder 

- Step 1: Find Root
- Step 2: Subproblem - Left Tree
- Step 3: Subproblem - Right Tree

8. (8%) Tree

- Preorder A B C H D E F G
- Inorder H C B D A F E G

8. (8%) Tree

- Preorder A B C H D E F G
- Inorder H C B D A F E G

8. (8%) Tree

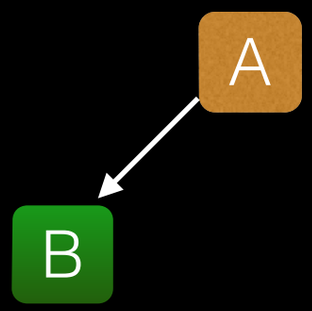
- Preorder B C H D E F G
- Inorder H C B D A F E G

8. (8%) Tree

- Preorder B C H D E F G
- Inorder H C B D A F E G

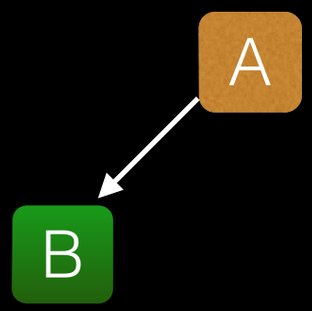
8. (8%) Tree

- Preorder C H D E F G
- Inorder H C B D A F E G



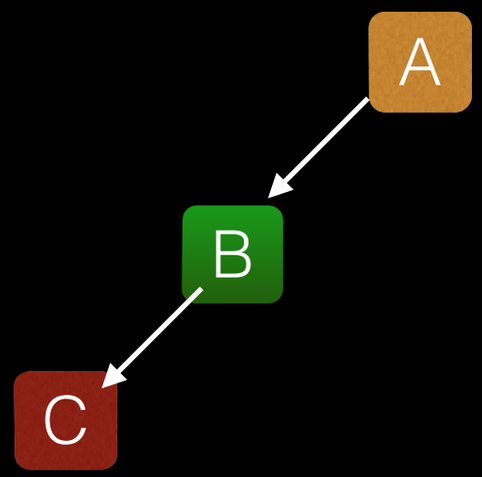
8. (8%) Tree

- Preorder C H D E F G
- Inorder H C B D A F E G



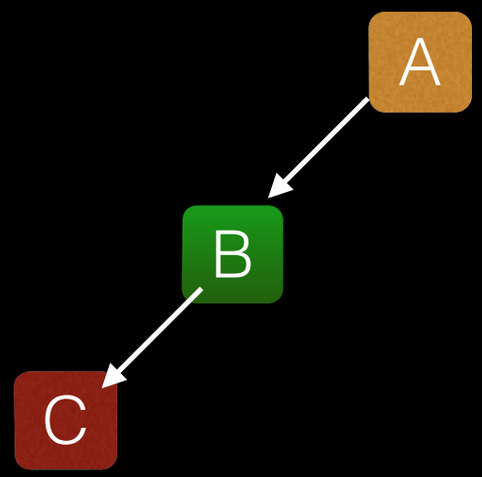
8. (8%) Tree

- Preorder H D E F G
- Inorder H C B D A F E G



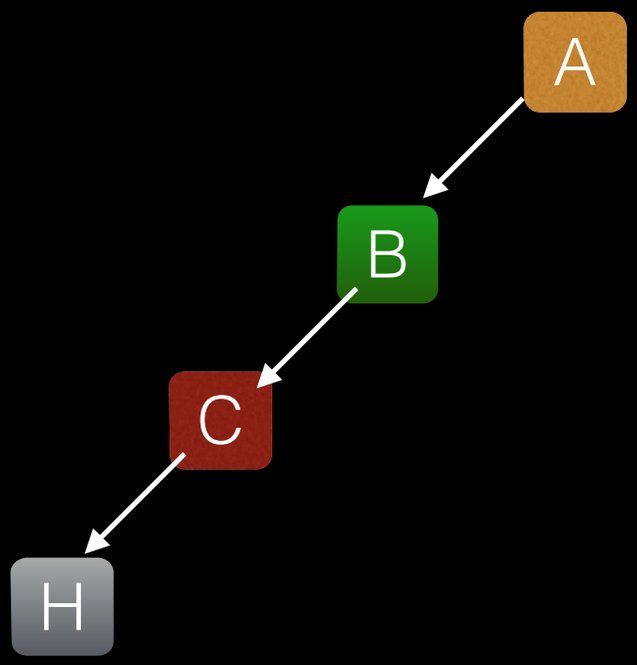
8. (8%) Tree

- Preorder H D E F G
- Inorder H C B D A F E G



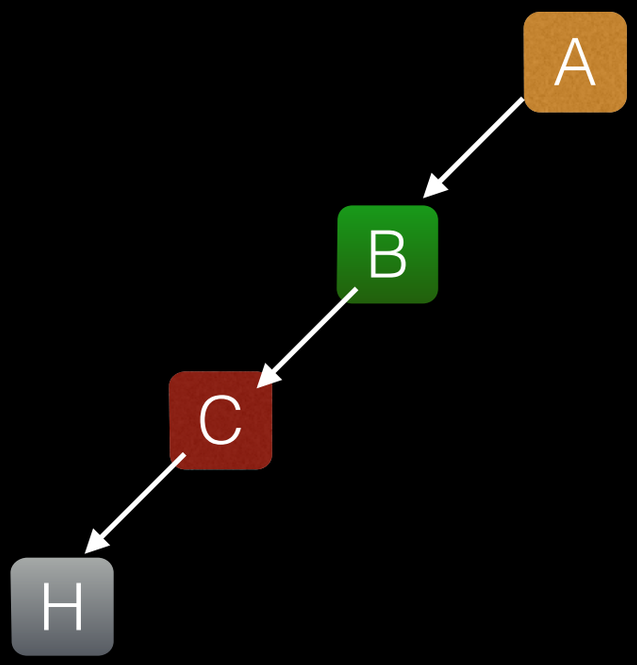
8. (8%) Tree

- Preorder D E F G
- Inorder H C B D A F E G



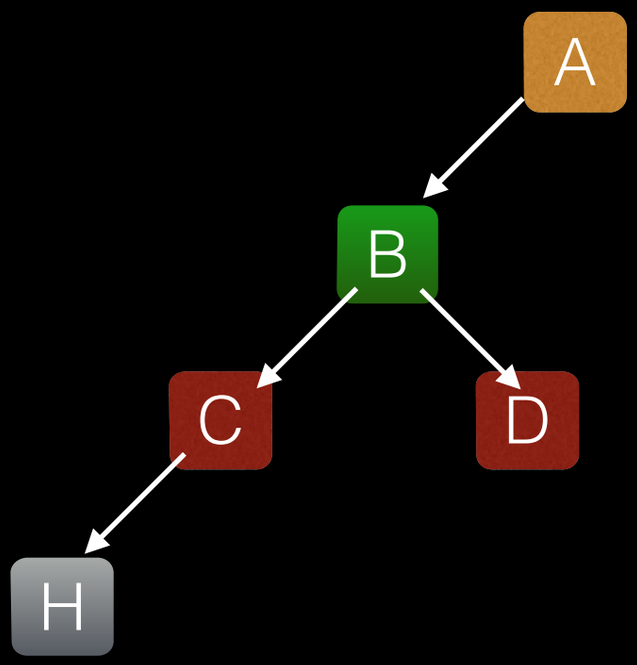
8. (8%) Tree

- Preorder D E F G
- Inorder H C B D A F E G



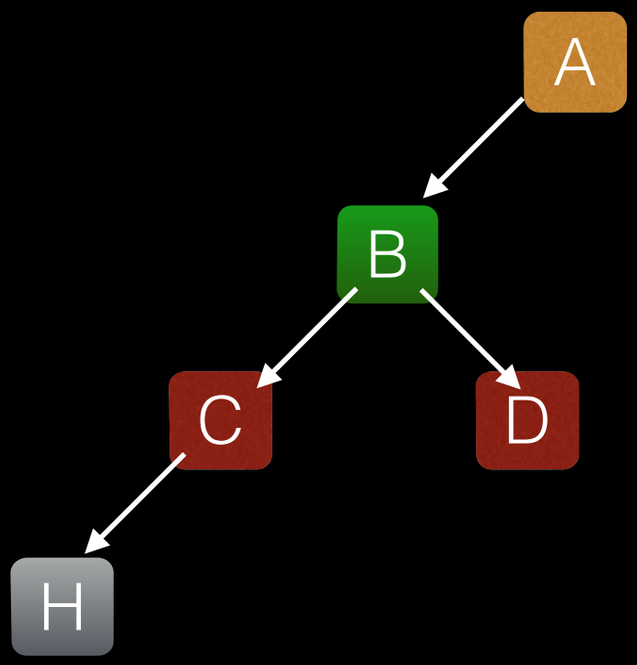
8. (8%) Tree

- Preorder E F G
- Inorder H C B D A F E G



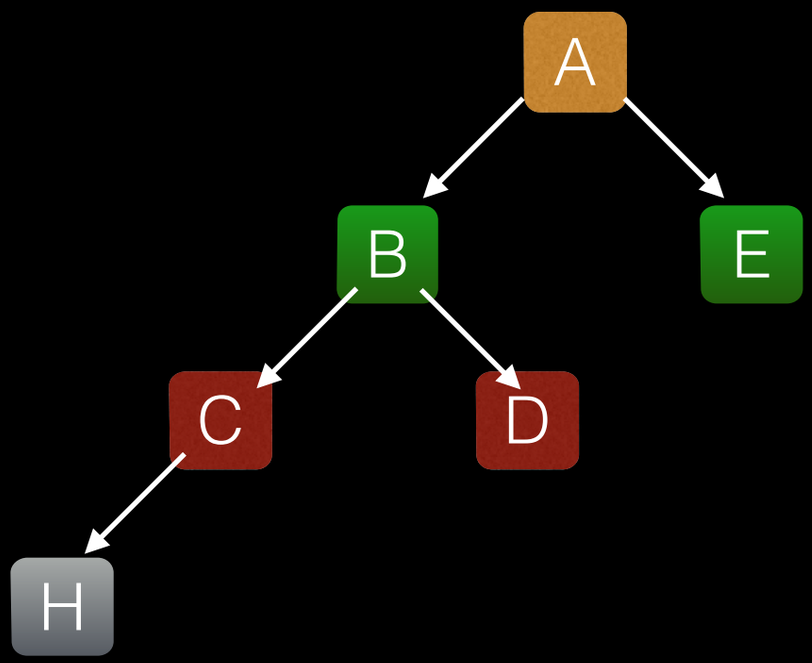
8. (8%) Tree

- Preorder E F G
- Inorder H C B D A F E G



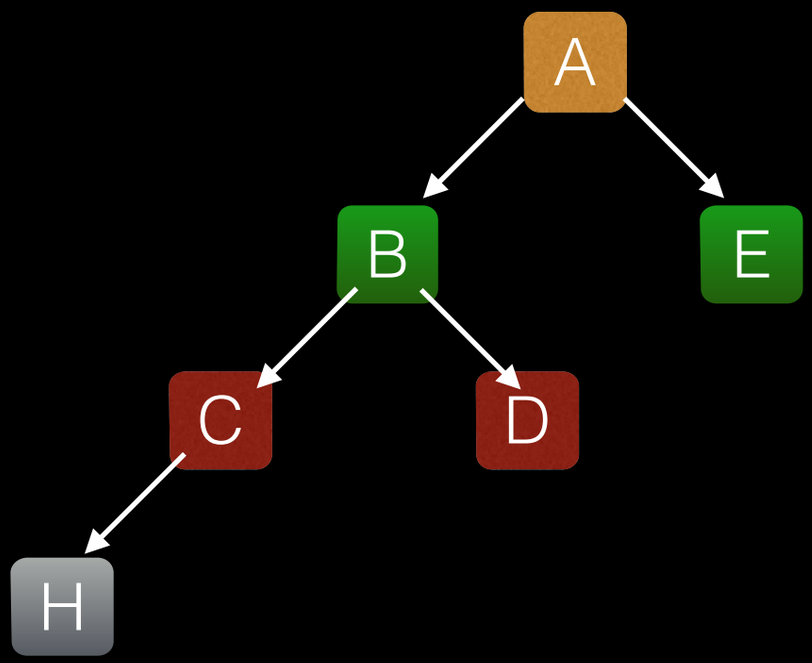
8. (8%) Tree

- Preorder F G
- Inorder H C B D A F E G



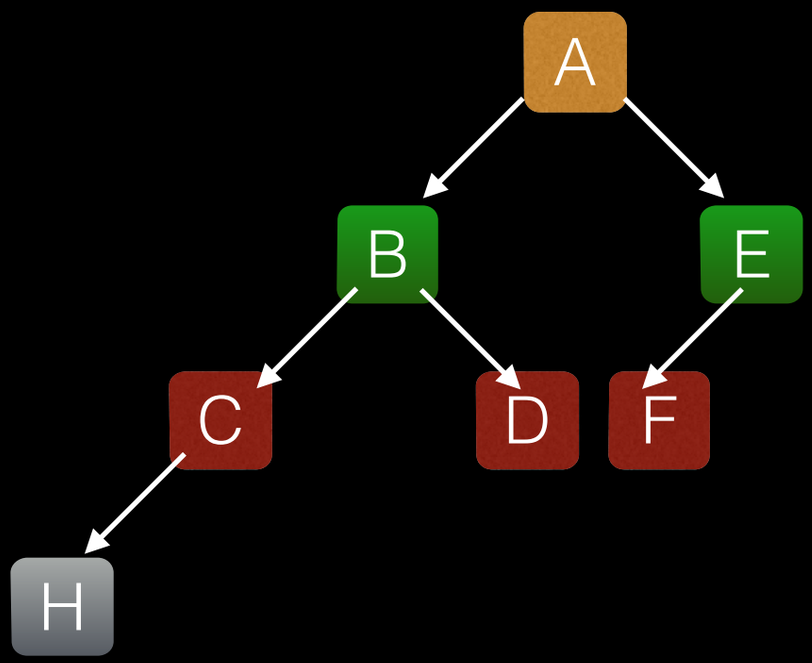
8. (8%) Tree

- Preorder F G
- Inorder H C B D A F E G



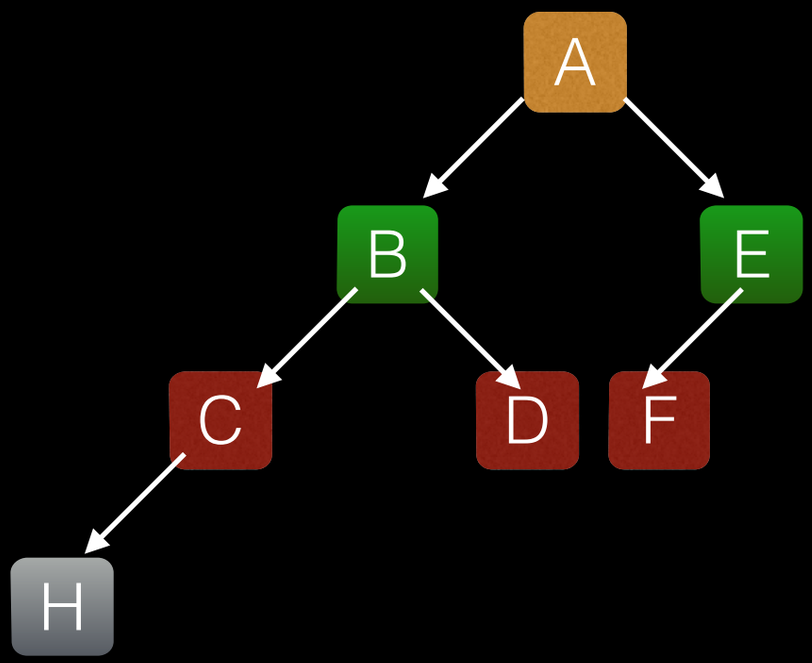
8. (8%) Tree

- Preorder G
- Inorder H C B D A F E G



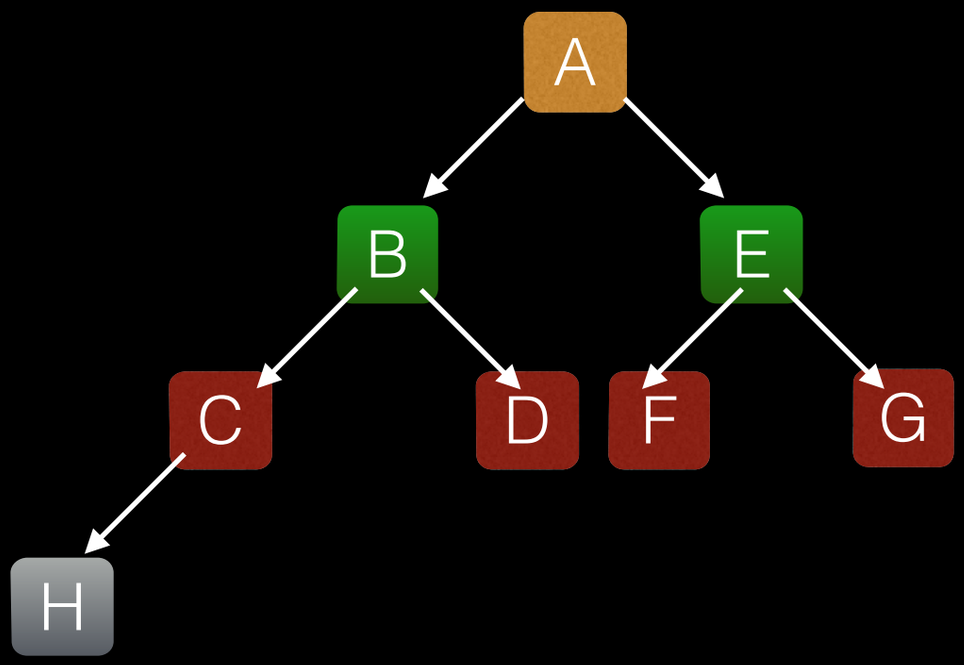
8. (8%) Tree

- Preorder G
- Inorder H C B D A F E G



8. (8%) Tree

- Preorder
- Inorder H C B D A F E G



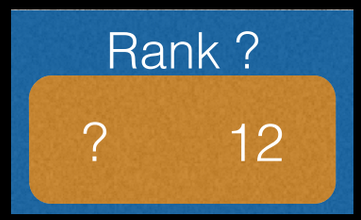
10. (8%) Tree

- An indexed binary search tree is a BST with each node containing an additional data field leftSize, which is one plus the number of nodes in the left subtree, and the rank of a node is its position in the inorder traversal.

- (a) Given a list of numbers {12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18},please orderly construct an indexed binary search tree and explicitly label the leftSize and the rank of each node.

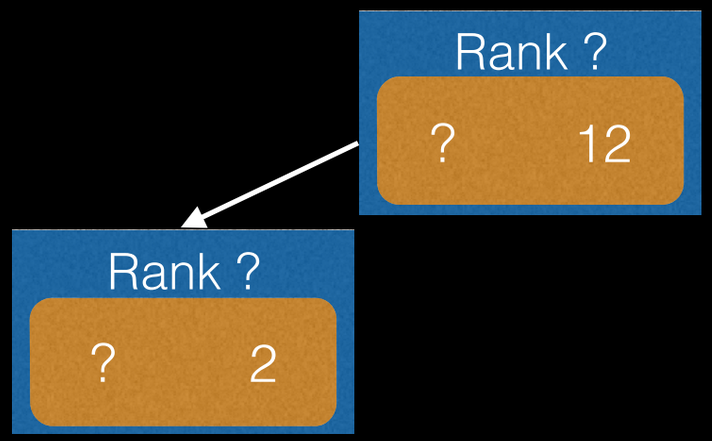
- (b) Show how to delete the seventh smallest element from the constructed indexed binary search tree.

10. (8%) Tree



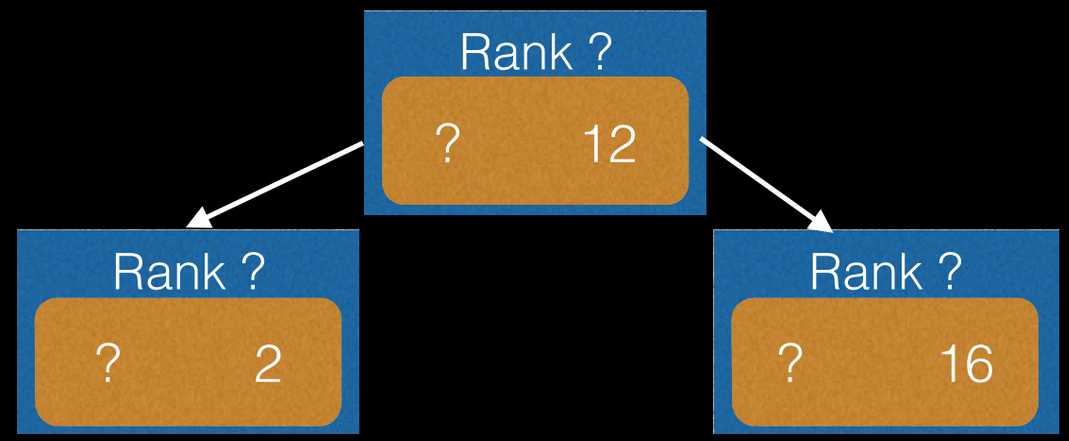
12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree



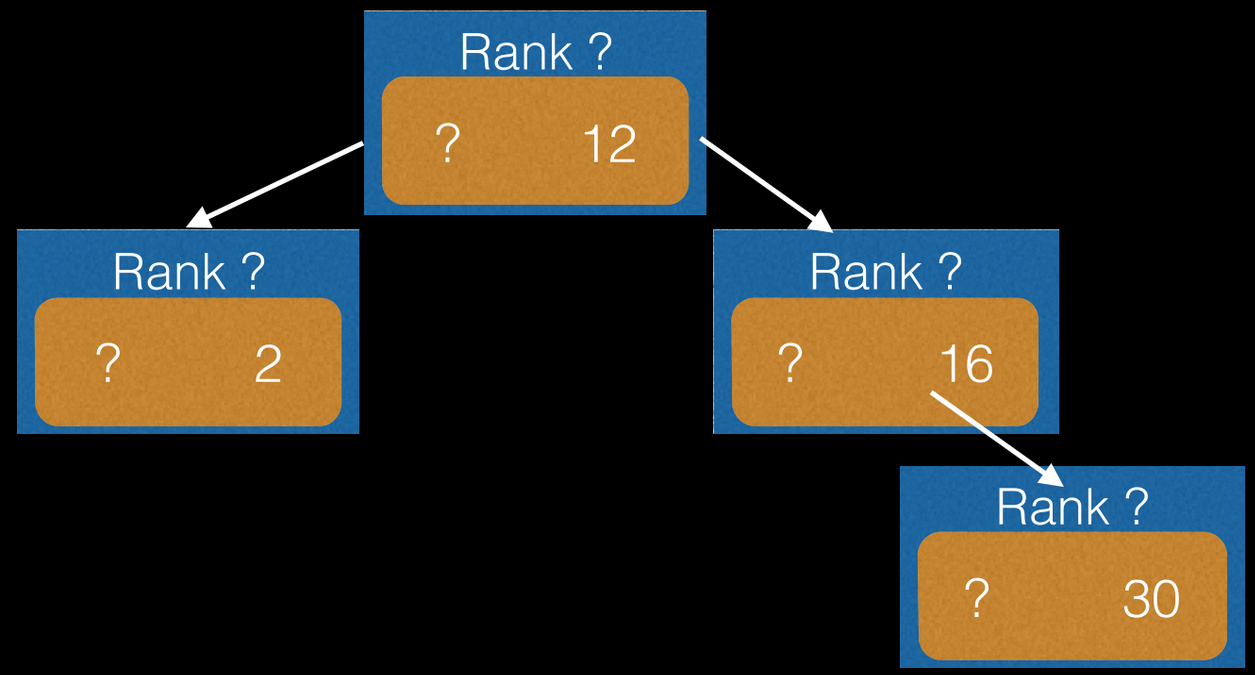
12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree

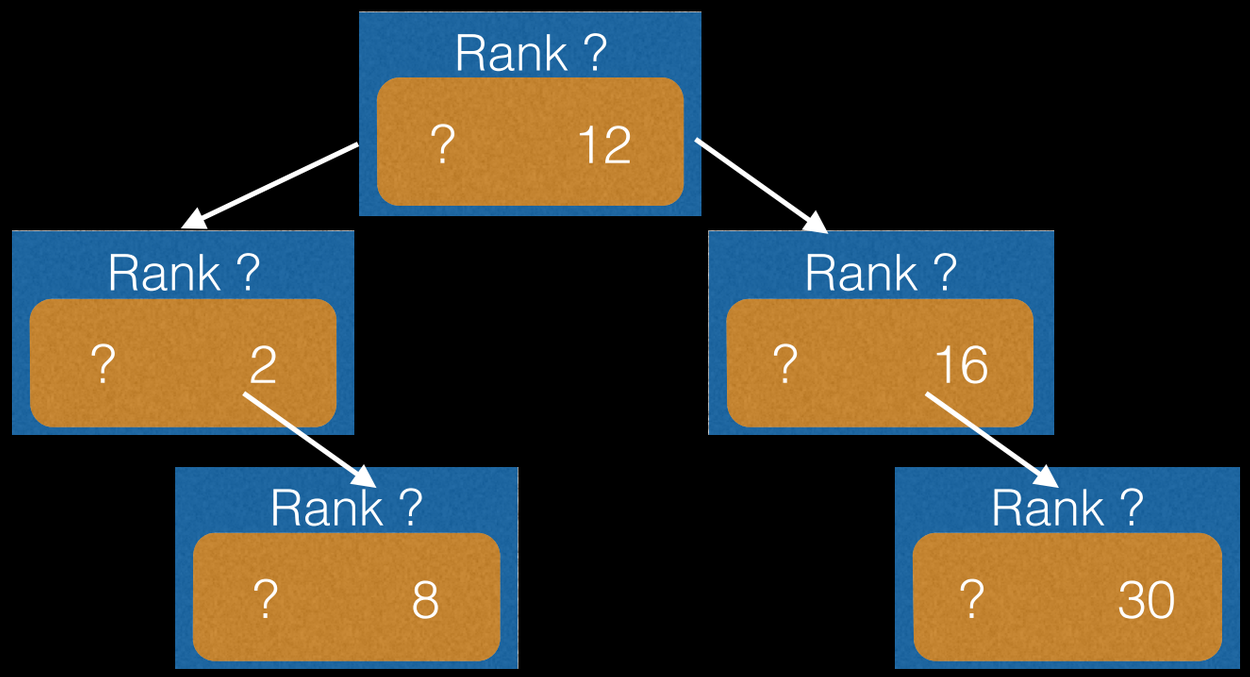


12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree

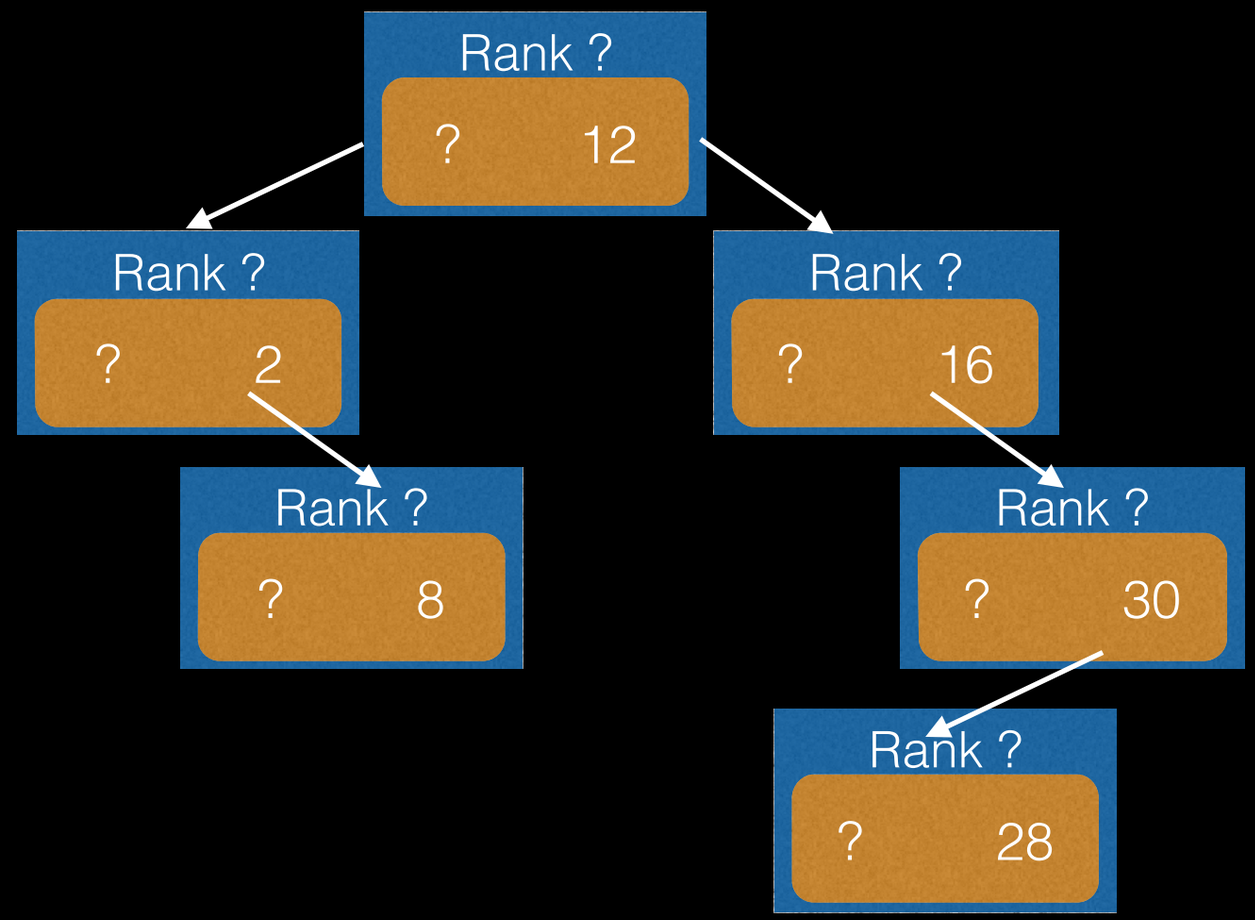


10. (8%) Tree



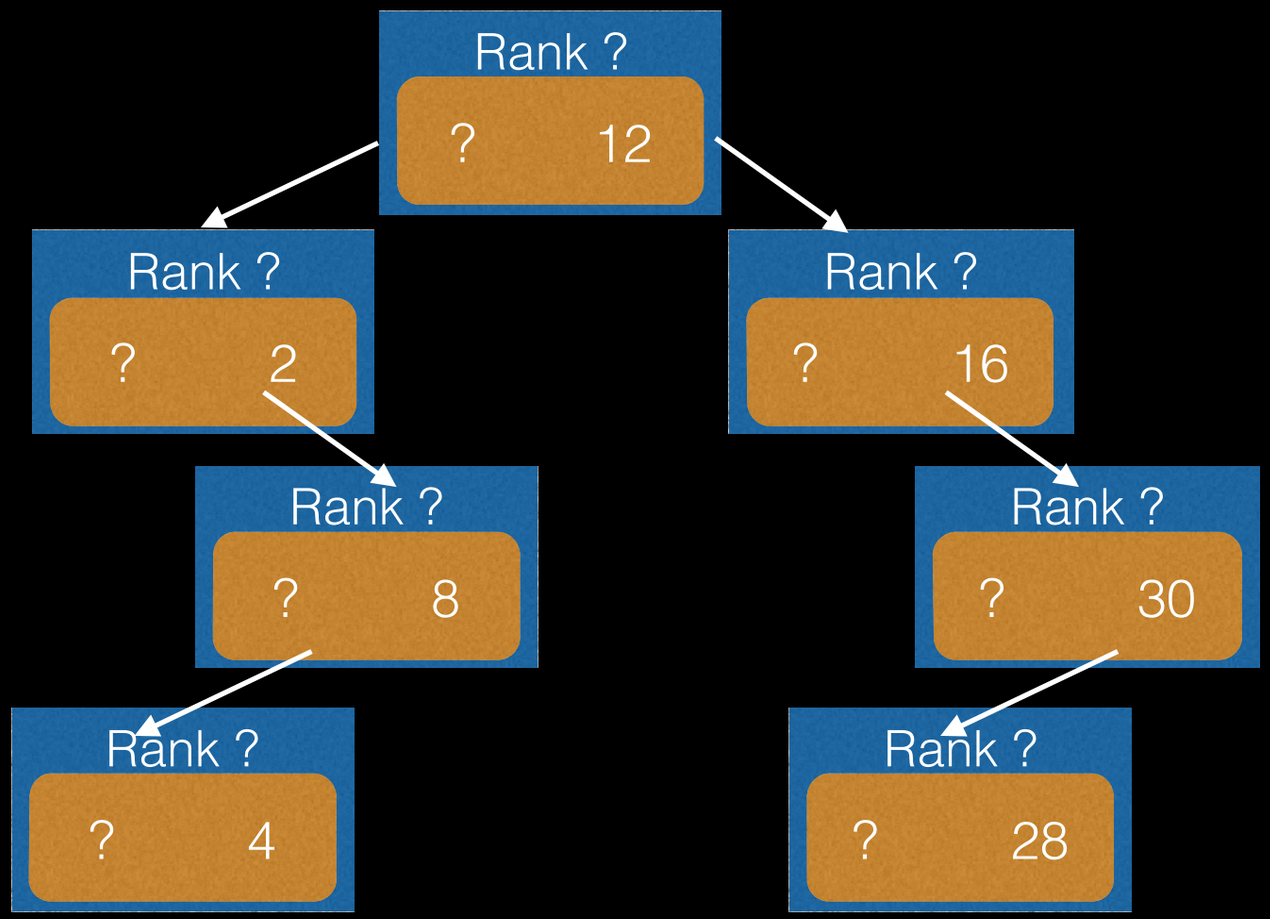
12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree



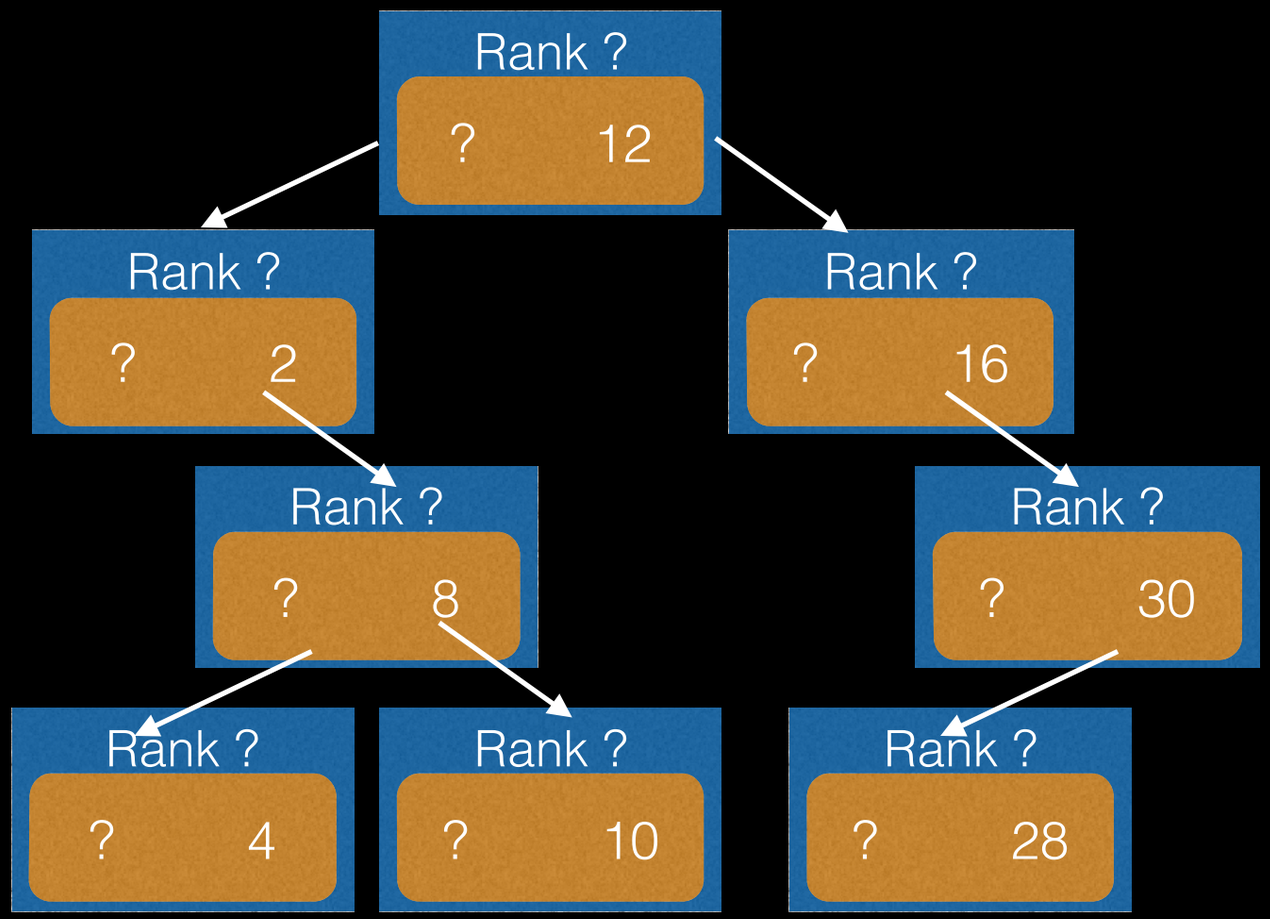
12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree



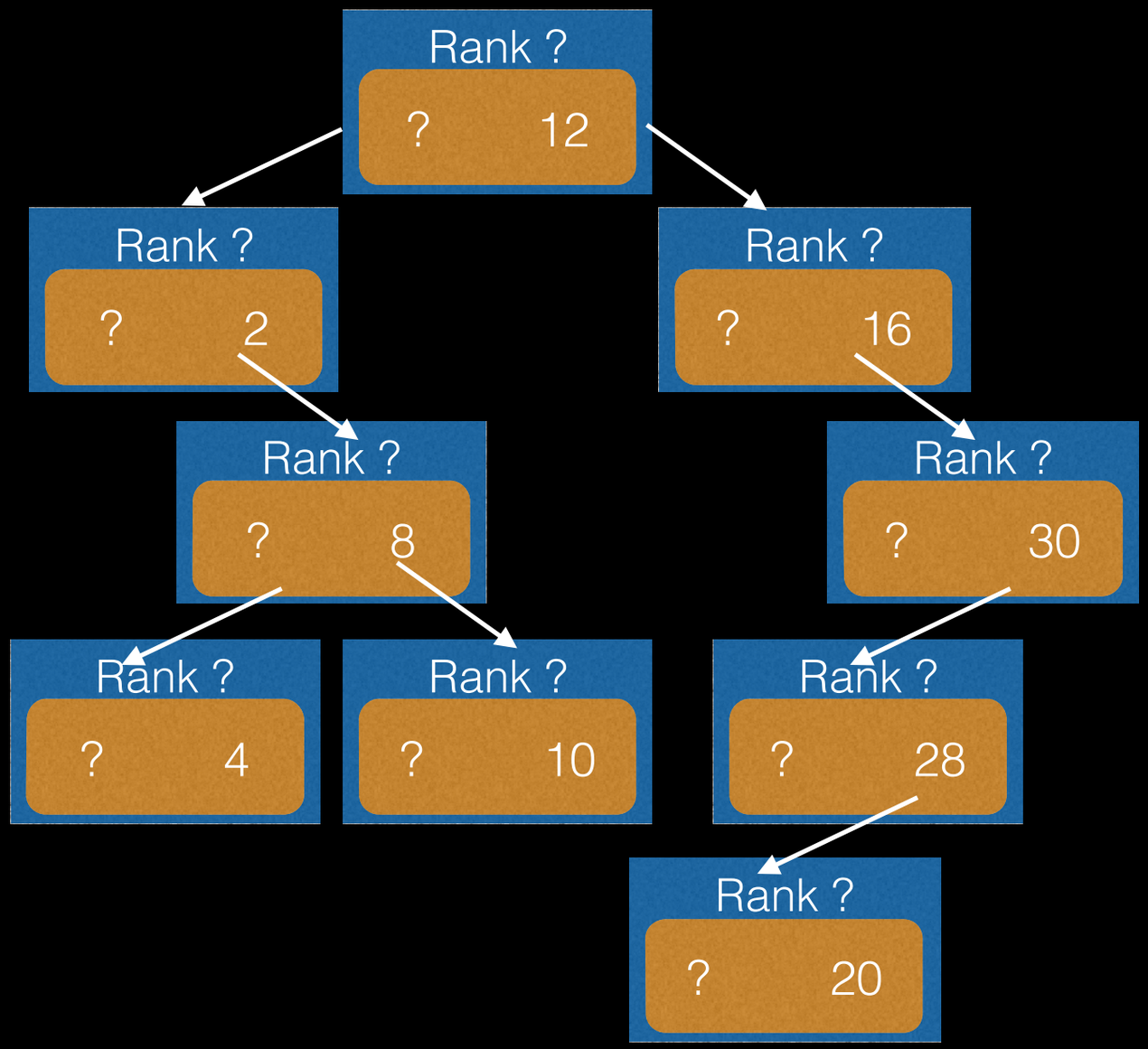
12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree



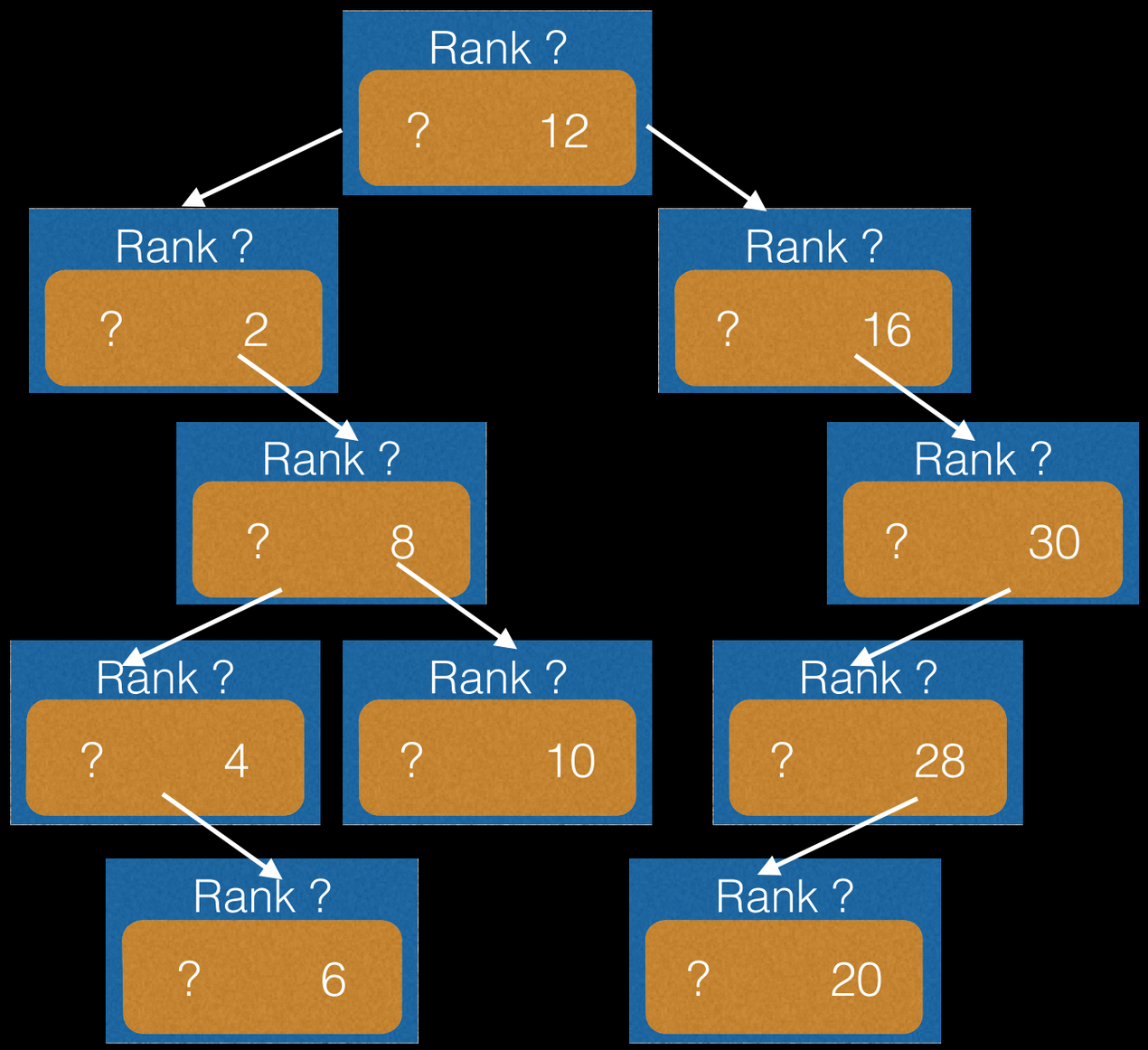
12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree



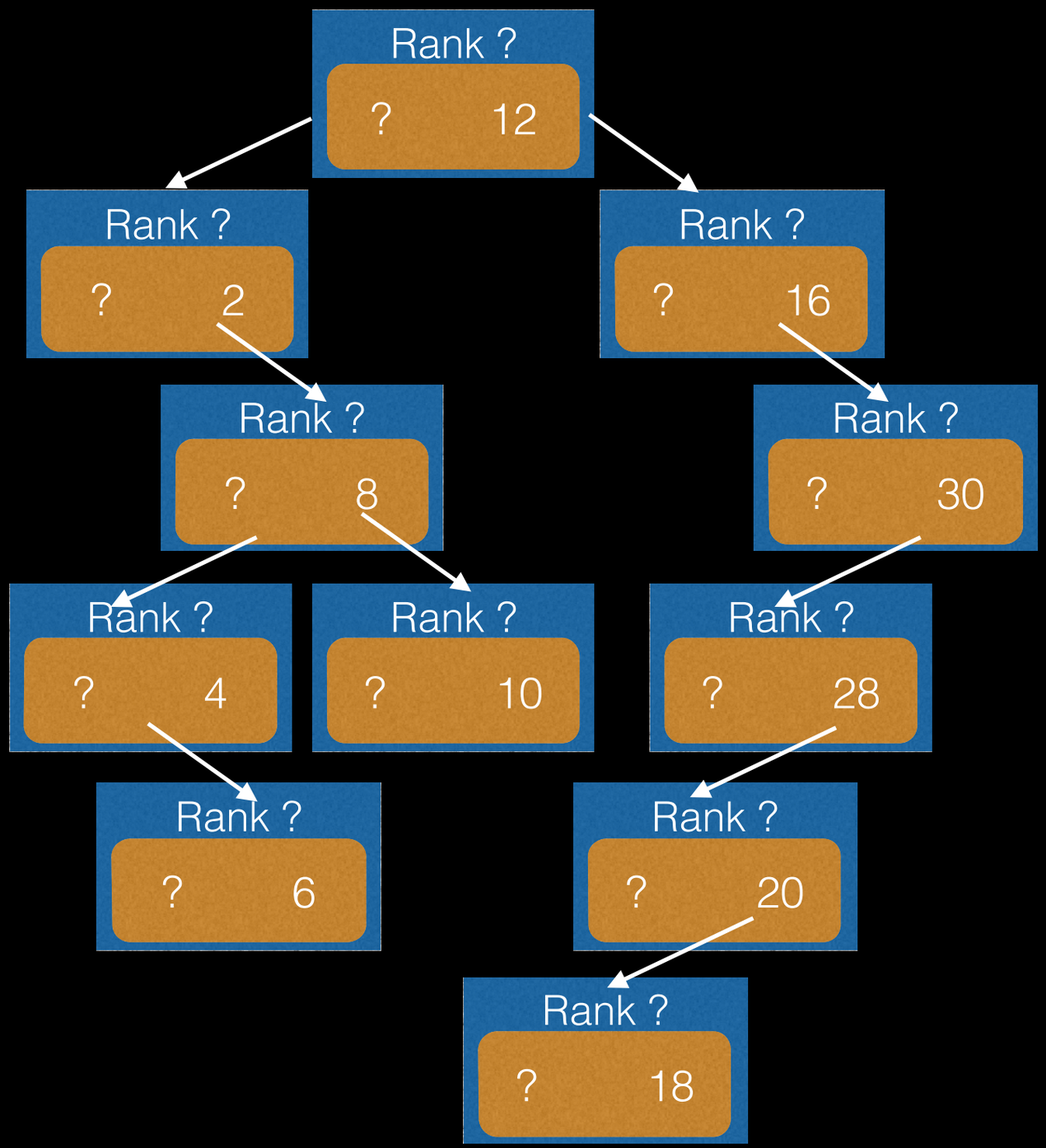
12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree



12	2	16	30	8	28	4	10	20	6	18

10. (8%) Tree



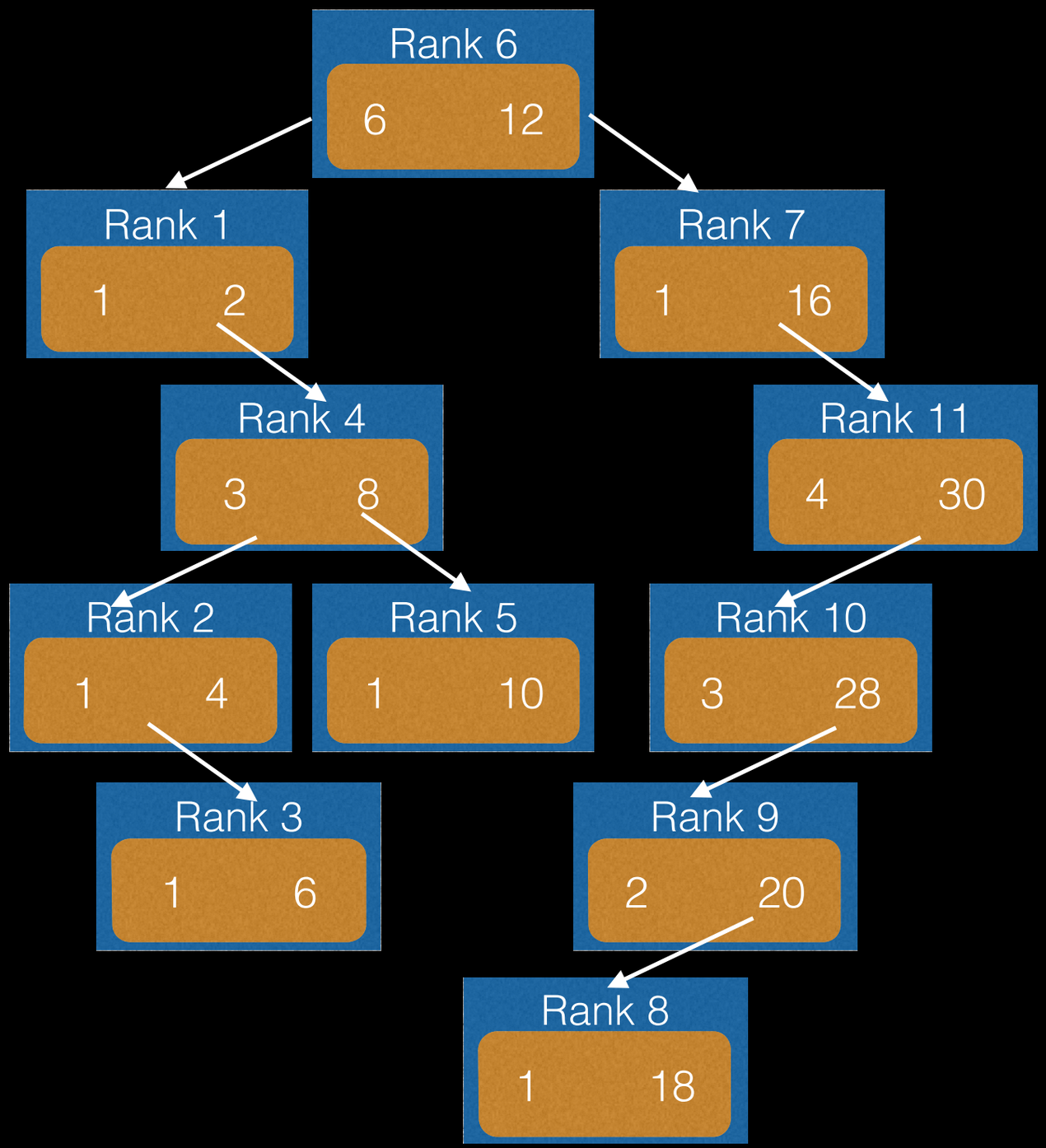
10. (8%) Tree

```
int countNode(Node* root)
{
  if(root==NULL)
   return 0;

  int leftSz  = countNode(root->left);
  int rightSz = countNode(root->right);
  root->leftSize = leftSz + 1;

  //return number of nodes
  return leftSz + rightSz + 1;
}
```

10. (8%) Tree

```
//calculate rank of each node with calculated leftSize
void calcRank(Node* root, int rankFrom)
{
  if(root->left)
     calcRank( root->left, rankFrom );
  if(root->right)
     calcRank( root->right, rankFrom+root->leftSize );

  //calculate rank
  root->rank = rankFrom + root->leftSize - 1;
}
calcRank(root, 1);
```

10. (8%) Tree


Rank 6
6
12
Rank 1
1
2
Rank 7
1
16
Rank 4
3
8
Rank 11
4
30
Rank 2
1
4
Rank 5
1
10
Rank 10
3
28
Rank 3
1
6
Rank 9
2
20
Rank 8
1
18

10. (8%) Tree

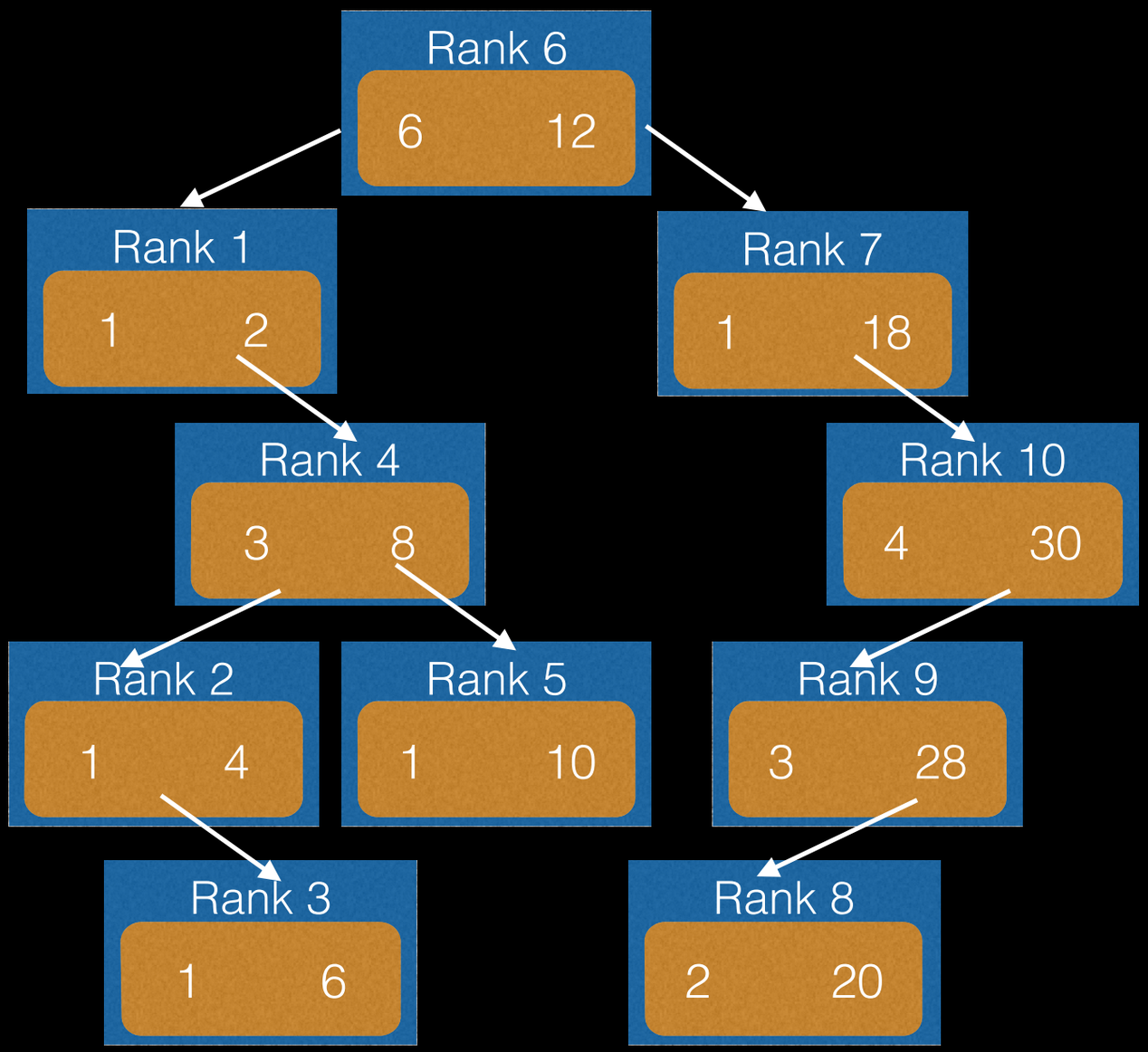
1. Find the Node
2. Delete the Node
3. Recalculate LeftSize & Rank

10. (8%) Tree

1. Find the Node

```
Node* FindNth(Node *root, int nth)
{
    if( root->leftSize == nth )
        return root;
    else if( root->leftSize > nth )
        return FindNth( root->leftTree, nth);
    else
        return FindNth( root->rightTree, nth-root->leftSize );
}
```

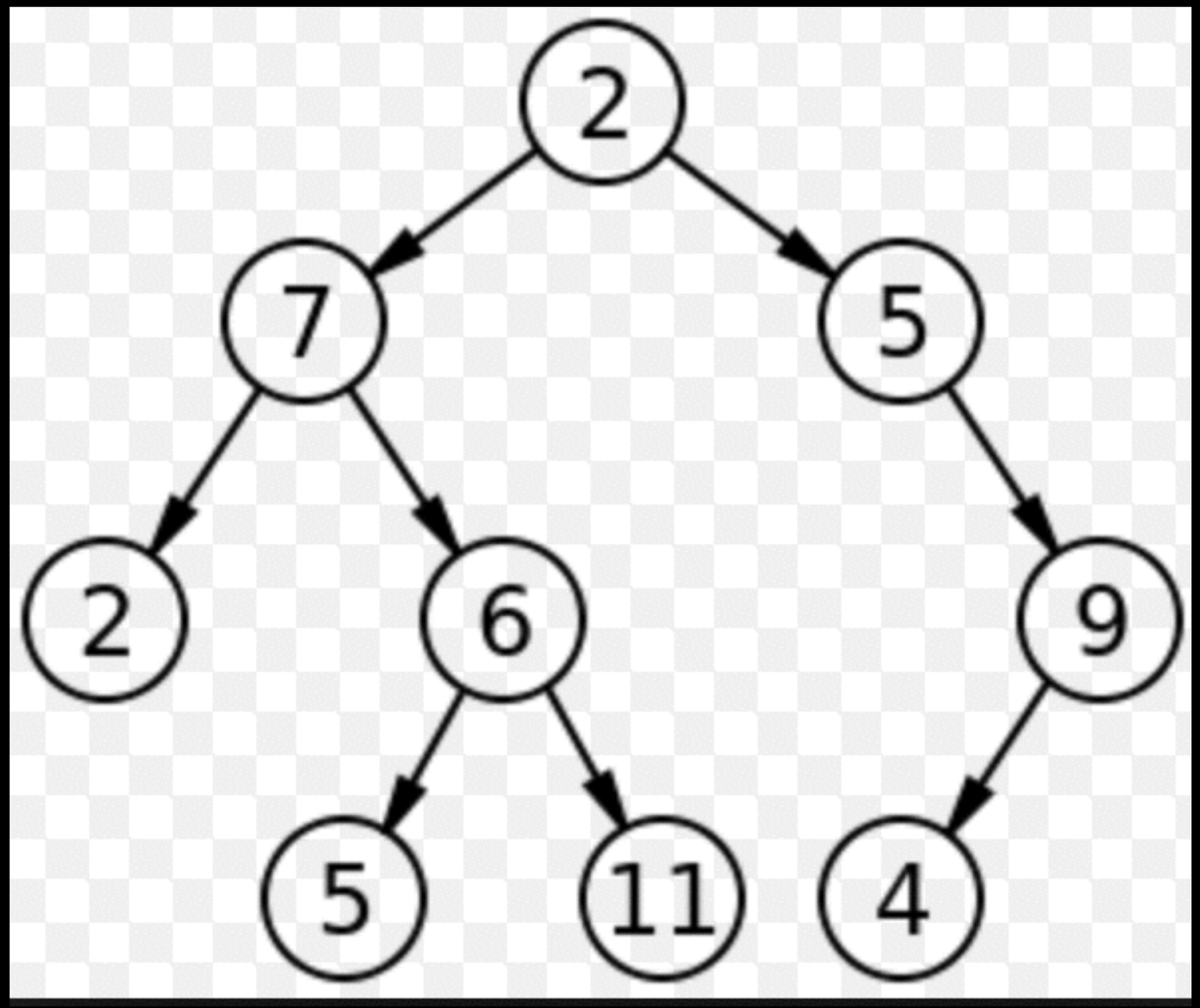
10. (8%) Tree


Rank 6
6
12
Rank 1
1
2
Rank 7
1
18
Rank 4
3
8
Rank 10
4
30
Rank 2
1
4
Rank 5
1
10
Rank 9
3
28
Rank 3
1
6
Rank 8
2
20

12. (10%) Tree

- A clocked tree is a Binary tree in which each node ni is associated with a non-negative delay, delay(ni). The path delay from a root to a node is defined as the summation of delay of all nodes along the path. The longestDelay is defined as the longest path delay among all root-to-leave(terminal node) paths. Write a C/C++ like recursive procedure:
  - *longestDelay(treenode *root, int AccumulatedDelay)*
- to compute the longest path delay "**MAX**".

11. (10%) Tree



11. (10%) Tree

```
int findLongestPath(treenode* root)
{
    if(root==NULL) return 0;

    int leftP  = findLongestPath(root->left );
    int rightP = findLongestPath(root->right);

    return root->delay + max(leftP, rightP);
}

MAX = findLongestPath(root);
```

11. (10%) Tree

```
findLongestPath(treenode* root, int AccumulatedDelay )
{
   if( root==NULL )return ;

   int upDelay = AccumulatedDelay + root->delay;
   MAX = max(MAX, upDelay);

   int leftP   = findLongestPath(root->left , upDelay);
   int rightP  = findLongestPath(root->right, upDelay);
}

MAX=-1;
findLongestPath(root, 0);
```