

## Sorting Algorithms

Sort [4,3,1,5,2] using selection sort, insertion sort and bubble sort

(From Kung-Hua Chang's practice5)

Which sorting algorithm could have produced the following array after 2 iterations?

Original sequence: 30 50 40 10 20 60 70 90 80 0

Sequence after 2 iterations: 30 10 20 40 50 60 70 0 80 90

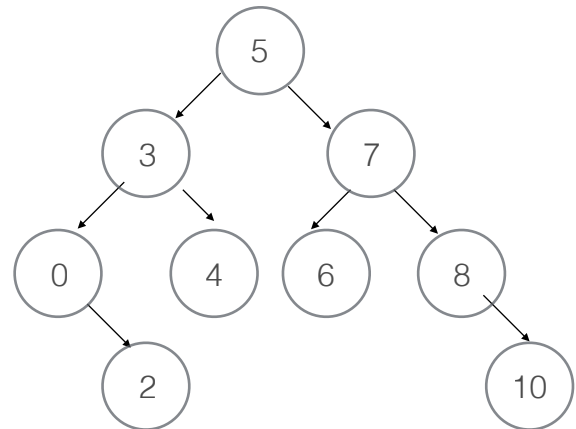
Sort [3,7,6,5,8,2,1,4] using merge sort and quick sort  
(Use the first element as the pivot for quick sort)

How to find the minimum k elements in an array?

How to find the median of an array?

## Tree

```
struct Node
{
    Node(const int &myVal) {
        value = myVal;
        left = right = nullptr;
    }
    int value;
    Node *left,*right;
};
class BinaryTree
{
public:
    BinaryTree() { m_root = nullptr; }
    ~BinaryTree() { freeTree(m_root); }
    void preorder(Node *node);
    void inorder(Node *node);
    void postorder(Node *node);
    void levelorder();
    void someorder();
    int numOfNonLeafNodes(Node *node);
    int height(Node *node);
private:
    Node *m_root;
};
```



What is the output for each of the following functions on the tree?

```
void BinaryTree::preorder(Node *node)
{
    if (node == nullptr) return;
    cout << node->value << " ";
    preorder(node->left);
    preorder(node->right);
}
```

```
void BinaryTree::inorder(Node *node)
{
    if (node == nullptr) return;
    inorder(node->left);
    cout << node->value << " ";
    inorder(node->right);
}
```

```

void BinaryTree::postorder(Node *node)
{
    if (node == nullptr) return;
    postorder(node->left);
    postorder(node->right);
    cout << node->value << " ";
}

void BinaryTree::levelorder()
{
    queue<Node*> q;
    q.push(m_root);
    while( ! q.empty() ) {
        Node *visited_node = q.front();
        q.pop();
        if(visited_node->left != nullptr )
            q.push(visited_node->left);
        if(visited_node->right!= nullptr )
            q.push(visited_node->right);
        cout << visited_node->value << " ";
    }
}

void BinaryTree::someorder()
{
    stack<Node*> q;
    q.push(m_root);
    while( ! q.empty() ) {
        Node *visited_node = q.top();
        q.pop();
        if(visited_node->right != nullptr )
            q.push(visited_node->right);
        if(visited_node->left!= nullptr )
            q.push(visited_node->left);
        cout << visited_node->value << " ";
    }
}

int BinaryTree::numOfNonLeafNodes(Node *node)
{
}

}

```

```
int BinaryTree::height(Node *node)
{

}

}
```

Given the preorder and inorder traversal of the tree, reconstruct it.  
preorder = {7,10,4,3,1,2,8,11}  
inorder = {4,10,3,1,7,11,8,2}