SORTING ALGORITHMS

1.) SELECTION SORT

CODE

```cpp
class Solution {
public:
    void selectionSort(vector<int>& arr) {
        int n = arr.size();

        for (int i = 0; i <= n - 2; i++) {
            int mini = i;
            for (int j = i; j <= n - 1; j++) {
                if (arr[j] < arr[mini]) {
                    mini = j;
                }
            }
            int temp = arr[mini];
            arr[mini] = arr[i];
            arr[i] = temp;
        }
    }
};
```

2.) BUBBLE SORT

CODE

```cpp
class Solution {
public:
    void bubbleSort(vector<int>& arr) {
        int n = arr.size();

        for(int i=n-1;i>=0;i--){
            for(int j=0;j<=i-1;j++){
                if(arr[j]>arr[j+1]){
                    int temp=arr[j+1];
                    arr[j+1]=arr[j];
                    arr[j]=temp;
                }
            }
        }
    }
};
```

SORTING ALGORITHMS

3.) INSERTION SORT

CODE

```
class Solution {
public:
    void insertionSort(vector<int>& arr) {
        int n = arr.size();

        for(int i=0;i<=n-1;i++){
            int j=i;
            while(j>0 && arr[j-1]>arr[j]){
                int temp=arr[j-1];
                arr[j-1]=arr[j];
                arr[j]=temp;
                j--;
            }
        }
    }
};
```

SORTING ALGORITHMS

4.) MERGE SORT

CODE

```
void merge(vector<int> &arr, int low, int mid, int high){
    vector<int> temp;
    int left=low;
    int right=mid+1;
    while(left<=mid && right<=high){
        if(arr[left]<=arr[right]){
            temp.push_back(arr[left]);
            left++;
        }
        else{
            temp.push_back(arr[right]);
            right++;
        }
    }

    while(left<=mid){
        temp.push_back(arr[left]);
        left++;
    }

    while(right<=high){
        temp.push_back(arr[right]);
        right++;
    }

    for(int i=low;i<=high;i++){
        arr[i]=temp[i-low];
    }
}

void ms(vector<int> &arr, int low, int high){
    if(low==high)
    return;
    int mid=(low+high)/2;
    ms(arr, low, mid);
    ms(arr, mid+1, high);
    merge(arr, low, mid, high);
}

void mergeSort(vector < int > & arr, int n) {
    ms(arr, 0, n-1);
}
```

SORTING ALGORITHMS

5.) QUICK SORT

CODE

```
#include <bits/stdc++.h>

int partition(vector<int> &arr, int low, int high){
    int pivot=arr[low];
    int i=low;
    int j=high;

    while(i<j){
        while(arr[i]<=pivot && i<=high-1){
            i++;
        }
        while(arr[j]>pivot && j>=low+1){
            j--;
        }

        if(i<j){
            swap(arr[i], arr[j]);
        }
    }

    swap(arr[low], arr[j]);
    return j;
}

void qs(vector<int> &arr, int low, int high){
    if(low<high){
        int pIndex=partition(arr, low, high);
        qs(arr, low, pIndex-1);
        qs(arr, pIndex+1, high);
    }
}

vector<int> quickSort(vector<int> arr)
{
    qs(arr, 0, arr.size()-1);
    return arr;
}
```

THANKYOU!