Dsa day 5  
  
1)BUBBLE SORT  
#include <bits/stdc++.h>

using namespace std;

void bubbleSort(vector<int>& arr) {

    int n = arr.size();

    bool swapped;

    for (int i = 0; i < n - 1; i++) {

        swapped = false;

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                swap(arr[j], arr[j + 1]);

                swapped = true;

            }

        }

              if (!swapped)

            break;

    }

}

void printVector(const vector<int>& arr) {

    for (int num : arr)

        cout << " " << num;

}

int main() {

    vector<int> arr = { 64, 34, 25, 12, 22, 11, 90 };

    bubbleSort(arr);

    cout << "Sorted array: \n";

    printVector(arr);

    return 0;

}

TC-O(N^2)

2)QUICK SORT

#include <bits/stdc++.h>

using namespace std;

int partition(vector<int>& arr, int low, int high) {

    int pivot = arr[high];

    int i = low - 1;

    for (int j = low; j <= high - 1; j++) {

        if (arr[j] < pivot) {

            i++;

            swap(arr[i], arr[j]);

        }

    }

    swap(arr[i + 1], arr[high]);

    return i + 1;

}

void quickSort(vector<int>& arr, int low, int high) {

    if (low < high) {

        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);

        quickSort(arr, pi + 1, high);

    }

}

int main() {

    vector<int> arr = {10, 7, 8, 9, 1, 5};

    int n = arr.size();

    quickSort(arr, 0, n - 1);

    for (int i = 0; i < n; i++) {

        cout << arr[i] << " ";

    }

    return 0;

}

OUTPUT:

TC-O(LOG N)

3) longest non repeating character substring

class Solution {

public:

    int lengthOfLongestSubstring(string s) {

        int maxlen=0;

        int left=0;

        int n=s.size();

        unordered\_set<char>charset;

        for(int right=0;right<n;right++){

            while(charset.find(s[right])!= charset.end()){

                charset.erase(s[left]);

                left++;

            }

            charset.insert(s[right]);

            maxlen=max(maxlen,right-left+1);

        }

        return maxlen;

    }

};

4)EDIT DISTANCE

class Solution {

public:

int minDistance(string word1, string word2) {

int n=word1.size();

int m=word2.size();

vector<vector<int>> dp(n+1,vector<int>(m+1,-1));

for(int i=0;i<=n;i++) dp[i][0]=i;

for(int j=0;j<=m;j++) dp[0][j]=j;

for(int i=1;i<=n;i++){

for(int j=1;j<=m;j++){

if(word1[i-1]==word2[j-1]) dp[i][j]= 0 + dp[i-1][j-1];

else dp[i][j]=1+min(dp[i-1][j],min(dp[i][j-1],dp[i-1][j-1]));

}

}

return dp[n][m];

}

};

5)K LARGEST ELEMENTS

class Solution {

public:

int findKthLargest(vector<int>& nums, int k) {

priority\_queue<int,vector<int>,greater<int>>pq;

for(int num:nums){

pq.push(num);

if(pq.size()>k){

pq.pop();

}

}

return pq.top();

}

};