SORTING

Algorithm	Best Time Complexity	Average Time Complexity	Worst Time Complexity	Worst Space Complexity
Linear Search	O(1)	O(n)	O(n)	O(1)
Binary Search	O(1)	O(log n)	O(log n)	O(1)
Bubble Sort	O(n)	O(n^2)	O(n^2)	O(1)
Selection Sort	O(n^2)	O(n^2)	O(n^2)	O(1)
Insertion Sort	O(n)	O(n^2)	O(n^2)	O(1)
Merge Sort	O(nlogn)	O(nlogn)	O(nlogn)	O(n)
Quick Sort	O(nlogn)	O(nlogn)	O(n^2)	O(log n)
Heap Sort	O(nlogn)	O(nlogn)	O(nlogn)	O(n)
Bucket Sort	O(n+k)	O(n+k)	O(n^2)	O(n)
Radix Sort	O(nk)	O(nk)	O(nk)	O(n+k)
Tim Sort	O(n)	O(nlogn)	O(nlogn)	O(n)
Shell Sort	O(n)	O((nlog(n))^2)	O((nlog(n))^2)	O(1)

1. SORT IN C++ STL

```
#include <iostream>
#include <algorithm>
using namespace std;

struct Point{
    int x,y;
};

bool MyComp(Point p1,Point p2){
    return p1.x<p2.x;
}
```

```
int main() {

    Point arr[]={{3,10},{2,8},{5,4}};

         int n=sizeof(arr)/sizeof(arr[0]);
         sort(arr,arr+n,MyComp);

         for(auto i: arr)
             cout<<i.x<<" "<<i.y<<endl;
}

#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {

    vector<int> v={10,20,5,7};

         sort(v.begin(),v.end());

         for(int x: v)
             cout<<x<<" ";

         sort(v.begin(),v.end(),greater<int>());

         cout<<endl;
         for(int x: v)
             cout<<x<<" ";
}

#include <iostream>
#include <algorithm>
using namespace std;
```

```
int main() {

        int arr[]={10,20,5,7};
        int n = sizeof(arr) / sizeof(arr[0]);

        sort(arr,arr+n);

        for(int x: arr)
            cout<<x<<" ";

        sort(arr,arr+n,greater<int>());

        cout<<endl;
        for(int x: arr)
            cout<<x<<" ";
}
```

2. BUBBLE SORT

```
#include<bits/stdc++.h>
using namespace std;

void bubbleSort(int arr[], int n){
    for(int i = 0;i < n; i++){
        for(int j = 0 ; j < n - i -1; j++){
            if( arr[j] > arr[j+1]){
                swap(arr[j], arr[j+1]);
            }
        }
    }
}

int main() {
        int a[] = {2, 1, 3, 4};
        bubbleSort(a, 4);
        for(int i = 0;i < 4; i++){
            cout<<a[i]<<" ";
        }
        return 0;
}
```

OPTIMIZED

```
#include<bits/stdc++.h>
using namespace std;

void bubbleSort(int arr[], int n){

    bool swapped;
    for(int i = 0;i < n; 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 == false)
        break;
    }
}

int main() {
        int a[] = {2, 1, 3, 4};
        bubbleSort(a, 4);
        for(int i = 0; i < 4; i++){
            cout<<a[i]<<" ";
        }
        return 0;
}
```

3. SELECTION SORT

```
#include<bits/stdc++.h>
using namespace std;

void selectionSort(int arr[], int n){

    for(int i = 0; i < n; i++){
        int min_ind = i;

        for(int j = i + 1; j < n; j++){
            if(arr[j] < arr[min_ind]){
                min_ind = j;
            }
        }

        swap(arr[i], arr[min_ind]);
    }


}

int main() {
        int a[] = {2, 1, 3, 4};
        selectionSort(a, 4);
        for(int i = 0;i < 4; i++){
            cout<<a[i]<<" ";
        }
        return 0;
}
```

4. INSERTION SORT

```
#include <iostream>
#include <algorithm>
using namespace std;

void iSort(int arr[],int n){

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

int main() {

    int arr[]={50,20,40,60,10,30};

         int n=sizeof(arr)/sizeof(arr[0]);
         iSort(arr,n);

         for(auto x: arr)
             cout<<x<<" ";
}
```

5. MERGE 2 SORTED ARRAYS

```
#include <iostream>
```

```
#include <algorithm>
using namespace std;

void merge(int a[], int b[], int m, int n){

    int i=0,j=0;
    while(i<m && j<n){
        if(a[i]<b[j])
            cout<<a[i++]<<" ";
        else
            cout<<b[j++]<<" ";
    }
    while(i<m)
        cout<<a[i++]<<" ";
    while(j<n)
        cout<<b[j++]<<" ";
}

int main() {

    int a[]={10,15,20,40};
    int b[]={5,6,6,10,15};

         int m=sizeof(a)/sizeof(a[0]);
         int n=sizeof(b)/sizeof(b[0]);
         merge(a,b,m,n);
}
```

6. MERGE FUNCTION

```
#include <iostream>
#include <algorithm>
using namespace std;
```

```
void merge(int arr[], int l, int m, int h){

    int n1=m-l+1, n2=h-m;
    int left[n1],right[n2];
    for(int i=0;i<n1;i++)
        left[i]=arr[i+l];
    for(int j=0;j<n2;j++)
        right[j]=arr[m+1+j];
    int i=0,j=0,k=l;
    while(i<n1 && j<n2){
        if(left[i]<=right[j])
            arr[k++]=left[i++];
        else
            arr[k++]=right[j++];
    }
    while(i<n1)
        arr[k++]=left[i++];
    while(j<n2)
        arr[k++]=right[j++];
}

int main() {

    int a[]={10,15,20,40,8,11,15,22,25};
         int l=0,h=8,m=3;

         merge(a,l,m,h);
         for(int x: a)
             cout<<x<<" ";
}
```

7. MERGE SORT

```
#include <iostream>
```

```
#include <algorithm>
using namespace std;

void merge(int arr[], int l, int m, int h){

    int n1=m-l+1, n2=h-m;
    int left[n1],right[n2];
    for(int i=0;i<n1;i++)
        left[i]=arr[i+l];
    for(int j=0;j<n2;j++)
        right[j]=arr[m+1+j];
    int i=0,j=0,k=l;
    while(i<n1 && j<n2){
        if(left[i]<=right[j])
            arr[k++]=left[i++];
        else
            arr[k++]=right[j++];
    }
    while(i<n1)
        arr[k++]=left[i++];
    while(j<n2)
        arr[k++]=right[j++];
}

void mergeSort(int arr[],int l,int r){
    if(r>l){
        int m=l+(r-l)/2;
        mergeSort(arr,l,m);
        mergeSort(arr,m+1,r);
        merge(arr,l,m,r);
    }
}

int main() {
    int a[]={10,5,30,15,7};
        int l=0,r=4;

        mergeSort(a,l,r);
        for(int x: a)
            cout<<x<<" ";
}
```

8. INTERSECTION OF 2 SORTED ARRAYS

```
#include <bits/stdc++.h>
using namespace std;

void intersection(int a[], int b[], int m, int n){
    int i=0,j=0;
    while(i<m && j<n){
        if(i>0 && a[i-1]==a[i]){
            i++;
            continue;
        }
        if(a[i]<b[j]){
            i++;
        }
        else if(a[i]>b[j]){
            j++;
        }
        else{
            cout<<a[i]<<" ";
            i++;j++;
        }
    }
}

int main() {
    int a[]={3,5,10,10,10,15,15,20};
    int b[]={5,10,10,15,30};

        int m=sizeof(a)/sizeof(a[0]);
        int n=sizeof(b)/sizeof(b[0]);
        intersection(a,b,m,n);
}
```

9. UNION OF 2 SORTED ARRAYS

```
#include <bits/stdc++.h>
using namespace std;

void printUnion(int a[], int b[], int m, int n){

    int i=0,j=0;
    while(i<m && j<n){
        if(i>0 && a[i-1]==a[i]){i++;continue;}
        if(j>0 && b[j-1]==b[j]){j++;continue;}
        if(a[i]<b[j]){cout<<a[i]<<" ";i++;}
        else if(a[i]>b[j]){cout<<b[j]<<" ";j++;}
        else{cout<<a[i]<<" ";i++;j++;}
    }
        while(i<m){if(i==0||a[i]!=a[i-1])cout<<a[i]<<" ";i++;}
        while(j<n){if(j==0||b[j]!=b[j-1])cout<<b[j]<<" ";j++;}
}

int main() {

    int a[]={3,8,8};
    int b[]={2,8,8,10,15};

        int m=sizeof(a)/sizeof(a[0]);
        int n=sizeof(b)/sizeof(b[0]);
        printUnion(a,b,m,n);
}
```

10. COUNT INVERSIONS IN AN ARRAY

```
#include <bits/stdc++.h>
using namespace std;

int countAndMerge(int arr[], int l, int m, int r)
{
    int n1=m-l+1, n2=r-m;
    int left[n1],right[n2];
    for(int i=0;i<n1;i++)
        left[i]=arr[i+l];
    for(int j=0;j<n2;j++)
        right[j]=arr[m+1+j];
    int res=0,i=0,j=0,k=l;
    while(i<n1 && j<n2){
        if(left[i]<=right[j])
            {arr[k++]=left[i++];}
        else{
            arr[k++]=right[j++];
            res=res+(n1-i);
        }
    }
    while(i<n1)
        arr[k++]=left[i++];
    while(j<n2)
        arr[k++]=right[j++];
    return res;
}

int countInv(int arr[], int l, int r)
```

```
{
    int res = 0;
    if (l<r) {

        int m = (r + l) / 2;

        res += countInv(arr, l, m);
        res += countInv(arr, m + 1, r);
        res += countAndMerge(arr, l, m , r);
    }
    return res;
}

int main() {

    int arr[]={2,4,1,3,5};

         int n=sizeof(arr)/sizeof(arr[0]);

         cout<<countInv(arr,0,n-1);
}
```

11. LOMUTO PARTITION

```
#include <bits/stdc++.h>
using namespace std;

int iPartition(int arr[], int l, int h)
{
    int pivot=arr[h];
    int i=l-1;
    for(int j=l;j<=h-1;j++){
        if(arr[j]<pivot){
            i++;
            swap(arr[i],arr[j]);
        }
    }
    swap(arr[i+1],arr[h]);
    return i+1;
}

int main() {

    int arr[]={10,80,30,90,40,50,70};

         int n=sizeof(arr)/sizeof(arr[0]);

         iPartition(arr,0,n-1);

         for(int x: arr)
             cout<<x<<" ";
}
```

12. QUICK SORT

```
#include <bits/stdc++.h>
using namespace std;

int iPartition(int arr[], int l, int h)
{
    int pivot=arr[h];
    int i=l-1;
    for(int j=l;j<=h-1;j++){
        if(arr[j]<pivot){
            i++;
            swap(arr[i],arr[j]);
        }
    }
    swap(arr[i+1],arr[h]);
    return i+1;
}

void qSort(int arr[],int l,int h){
    if(l<h){
        int p=iPartition(arr,l,h);
        qSort(arr,l,p-1);
        qSort(arr,p+1,h);
    }
}

int main() {

    int arr[]={8,4,7,9,3,10,5};

        int n=sizeof(arr)/sizeof(arr[0]);

        qSort(arr,0,n-1);

        for(int x: arr)
            cout<<x<<" ";
}
```

13. KTH SMALLEST ELEMENT

```
#include <bits/stdc++.h>
using namespace std;

int partition(int arr[], int l, int h)
{
    int pivot=arr[h];
    int i=l-1;
    for(int j=l;j<=h-1;j++){
        if(arr[j]<pivot){
            i++;
            swap(arr[i],arr[j]);
        }
    }
    swap(arr[i+1],arr[h]);
    return i+1;
}

int kthSmallest(int arr[],int n,int k){
    int l=0,r=n-1;
    while(l<=r){
        int p=partition(arr,l,r);
        if(p==k-1)
            return p;
        else if(p>k-1)
            r=p-1;
        else
            l=p+1;
    }
    return -1;
}

int main() {

    int arr[]={10,4,5,8,11,6,26};

         int n=sizeof(arr)/sizeof(arr[0]);int k=5;

         int index=kthSmallest(arr,n,k);

         cout<<arr[index];

}
```

14. CHOCOLATE DISTRIBUTION PROBLEM

```
#include <bits/stdc++.h>
using namespace std;

int minDiff(int arr[],int n,int m){
    if(m>n)
        return -1;
    sort(arr,arr+n);
    int res=arr[m-1]-arr[0];
    for(int i=0;(i+m-1)<n;i++)
        res=min(res,arr[i+m-1]-arr[i]);
    return res;
}

int main() {

    int arr[]={7,3,2,4,9,12,56};

        int n=sizeof(arr)/sizeof(arr[0]);int m=3;

        cout<<minDiff(arr,n,m);
}
```

15. SORT AN ARRAY WITH 2 TYPES OF ELEMENTS

```
#include <bits/stdc++.h>
using namespace std;

void sort(int arr[],int n){
    int i=-1,j=n;
    while(true)
    {
        do{i++;}while(arr[i]<0);
        do{j--;}while(arr[j]>=0);
        if(i>=j)return;
        swap(arr[i],arr[j]);
    }
}

int main() {

    int arr[]={13,-12,18,-10};

         int n=sizeof(arr)/sizeof(arr[0]);

         sort(arr,n);

         for(int x:arr)
             cout<<x<<" ";
}
```

16. SORT AN ARRAY WITH 3 TYPES OF ELEMENTS

```
#include <bits/stdc++.h>
using namespace std;

void sort(int arr[],int n){
    int l=0,h=n-1,mid=0;
    while(mid<=h){
        switch(arr[mid]){
            case 0:
                swap(arr[l],arr[mid]);
                l++;mid++;
                break;
            case 1:
                mid++;
                break;
            case 2:
                swap(arr[h],arr[mid]);
                h--;
                break;
        }
    }

}

int main() {

    int arr[]={0,1,1,2,0,1,1,2};

         int n=sizeof(arr)/sizeof(arr[0]);

         sort(arr,n);

         for(int x:arr)
             cout<<x<<" ";
}
```

17. MINIMUM DIFFERENCE IN AN ARRAY

```
#include <bits/stdc++.h>
using namespace std;


int getMinDiff(int arr[], int n){
    sort(arr, arr + n);
    int res = INT_MAX;

    for(int i = 1; i < n; i++){
        res = min(res, arr[i] - arr[i-1]);
    }
    return res;
}

int main() {

        int n;
        cin>>n;
        int arr[n];
        for(int i = 0; i < n; i++){
            cin>>arr[i];
        }

        cout<<getMinDiff(arr, n );
        return 0;
}
```

18. MERGE OVERLAPPING INTERVALS

```
#include<bits/stdc++.h>
using namespace std;

struct Interval
{
    int s, e;
};

bool mycomp(Interval a, Interval b)
{ return a.s < b.s; }

void mergeIntervals(Interval arr[], int n)
{
    sort(arr, arr+n, mycomp);

    int res = 0;
    for (int i=1; i<n; i++)
    {
        if (arr[res].e >=  arr[i].s)
        {
            arr[res].e = max(arr[res].e, arr[i].e);
            arr[res].s = min(arr[res].s, arr[i].s);
        }
        else {
            res++;
            arr[res] = arr[i];
        }
    }

    for (int i = 0; i <= res; i++)
        cout << "[" << arr[i].s << ", " << arr[i].e << "] ";
}

int main()
{
    Interval arr[] =  { {5,10}, {3,15}, {18,30}, {2,7} };
    int n = sizeof(arr)/sizeof(arr[0]);
    mergeIntervals(arr, n);

    return 0;
}
```

19. MEETING THE MAXIMUM GUESTS

```
#include<bits/stdc++.h>
using namespace std;

int maxGuest(int arr[],int dep[],int n)
{
    sort(arr, arr+n);
    sort(dep, dep+n);

    int i=1,j=0,res=1,curr=1;
    while(i<n && j<n){
        if(arr[i]<dep[j]){
            curr++;i++;
        }
        else{
            curr--;j++;
        }
        res=max(curr,res);
    }
   return res;
}

int main()
{
    int arr[] = { 900, 600, 700};
    int dep[] = { 1000, 800, 730};
    int n = sizeof(arr)/sizeof(arr[0]);

    cout<<maxGuest(arr,dep, n);

    return 0;
}
```

20. CYCLE SORT

```
#include<bits/stdc++.h>
using namespace std;

void cycleSortDistinct(int arr[], int n)
{
    for(int cs=0;cs<n-1;cs++){
        int item=arr[cs];
        int pos=cs;
        for(int i=cs+1;i<n;i++)
            if(arr[i]<item)
                pos++;
        swap(item,arr[pos]);
        while(pos!=cs){
            pos=cs;
            for(int i=cs+1;i<n;i++)
                if(arr[i]<item)
                    pos++;
            swap(item,arr[pos]);
        }
    }
}

int main()
{
    int arr[] = { 20,40,50,10,30 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cycleSortDistinct(arr, n);

    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```

21. HEAP SORT

```
#include <iostream>
using namespace std;
```

```
void heapify(int arr[], int n, int i)
{
        int largest = i;
        int l = 2*i + 1;
        int r = 2*i + 2;
        if (l < n && arr[l] > arr[largest])
                largest = l;

        if (r < n && arr[r] > arr[largest])
                largest = r;

        if (largest != i)
        {
                swap(arr[i], arr[largest]);
                heapify(arr, n, largest);
        }
}

void buildheap(int arr[],int n){
    for (int i = n / 2 - 1; i >= 0; i--)
                heapify(arr, n, i);
}
void heapSort(int arr[], int n)
{
        buildheap(arr,n);

        for (int i=n-1; i>0; i--)
        {
                swap(arr[0], arr[i]);
                heapify(arr, i, 0);
        }
}

void printArray(int arr[], int n)
{
        for (int i=0; i<n; ++i)
                cout << arr[i] << " ";
        cout << "\n";
}

int main()
{
        int arr[] = {12, 11, 13, 5, 6, 7};
        int n = sizeof(arr)/sizeof(arr[0]);

        heapSort(arr, n);

        cout << "Sorted array is \n";
        printArray(arr, n);
}
```

22. COUNTING SORT

```
#include<bits/stdc++.h>
using namespace std;

void countSort(int arr[], int n, int k)
{
    int count[k];
    for(int i=0;i<k;i++)
        count[i]=0;
    for(int i=0;i<n;i++)
        count[arr[i]]++;

    for(int i=1;i<k;i++)
        count[i]=count[i-1]+count[i];

    int output[n];
    for(int i=n-1;i>=0;i--){
        output[count[arr[i]]-1]=arr[i];
        count[arr[i]]--;
    }
    for(int i=0;i<n;i++)
        arr[i]=output[i];

}

int main()
{
    int arr[] = { 1,4,4,1,0,1 };
    int n = sizeof(arr) / sizeof(arr[0]);
    int k=5;
    countSort(arr, n, k);

    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```

23. RADIX SORT

```
#include<bits/stdc++.h>
using namespace std;

void countingSort(int arr[], int n, int exp)
{
    int output[n];
    int count[10] = { 0 };
    for (int i = 0; i < n; i++)
        count[(arr[i] / exp) % 10]++;

    for (int i = 1; i < 10; i++)
```

```
        count[i] += count[i - 1];

    for (int i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }

    for (int i = 0; i < n; i++)
        arr[i] = output[i];
}

void radixsort(int arr[], int n)
{
    int mx = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > mx)
            mx = arr[i];

    for (int exp = 1; mx / exp > 0; exp *= 10)
        countingSort(arr, n, exp);
}

int main()
{
    int arr[] = { 319,212,6,8,100,50 };
    int n = sizeof(arr) / sizeof(arr[0]);
    radixsort(arr, n);

    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```

24. BUCKET SORT

```
#include<bits/stdc++.h>
using namespace std;

void bucketSort(int arr[], int n, int k)
{
    int max_val=arr[0];
    for(int i=1;i<n;i++)
        max_val=max(max_val,arr[i]);

    max_val++;
    vector<int> bkt[k];

    for (int i = 0; i < n; i++) {
        int bi = (k * arr[i])/max_val;
        bkt[bi].push_back(arr[i]);
    }

    for (int i = 0; i < k; i++)
        sort(bkt[i].begin(), bkt[i].end());

    int index = 0;
    for (int i = 0; i < k; i++)
        for (int j = 0; j < bkt[i].size(); j++)
            arr[index++] = bkt[i][j];
}



int main()
{
    int arr[] = { 30,40,10,80,5,12,70 };
    int n = sizeof(arr) / sizeof(arr[0]); int k=4;
    bucketSort(arr, n, k);

    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```