1.Write program to implement Binary Search

```
#include <bits/stdc++.h>
using namespace std;
int binarysearch(int arr[], int x, int low, int high)
{
   while (low <= high)
   {
      int mid = (low + high) / 2;
      if (arr[mid] == x)
      return mid;
      if (arr[mid] < x)
         low = mid + 1;
      else
         high = mid - 1;
   }
   return -1;
}
int main()
{
   int n;
   cout << "enter array size";
   cin >> n;
   int key;
   int a[n];

   cout << "enter Array elements ";
   for (int i = 0; i < n; i++)
   {
      cin >> a[i];
   }
   cout << "Enter element to search ";
   cin >> key;
   int result = binarysearch(a, key, 0, n - 1);
   if (result==-1)
      cout << "Element not found";
   else
      cout << "element found at index " << result;

   return 0;
}
```

Output:

```
enter array size 5

enter Array elements 3 5 7 8 9

Enter element to search 5

element found at index 1
```

2.Write program in C to implement Bubble Sort.

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

using namespace std;

int main() {

   int n;

   cout<<"enter size of array";cin>>n;

   int a[n];

   cout<<"enter array elments ";

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

   {

      cin>>a[i];

   }

   int counter=1;

   while(counter<n)

   {

      for(int i=0;i<n-counter;i++)

      {

         if(a[i]>a[i+1])

         {

            int temp=a[i];

            a[i]=a[i+1];

            a[i+1]=temp;

         }

      }

      counter++;

   }

   cout<<"Sorted array is ";
```

```
    for(int i=0;i<n;i++)

    {

        cout<<a[i]<<" ";

    }

    return 0;

}
```

```
Output:
enter size of array5

enter array elments 4 7 98 34 65

Sorted array is 4 7 34 65 98
```

3. Write program in C to implement Linear Search

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

using namespace std;

int main() {

    int n;  cout<<"enter array size";

    cin>>n;

     int key,pos=-1;

    int a[n];

   cout<<"enter Array elements ";

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

    {

        cin>>a[i];

    }

cout<<"Enter element to search ";

cin>>key;

for(int i=0;i<n;i++)
{
    if(a[i]==key)
    {
        pos=i;
    }


}
if(pos==-1)
{
cout<<"element not found";
}
else{
    cout<<"elment found at position "<<pos;
}
    return 0;
}
```

Output:

```
enter array size5
enter Array elements 6 8 45 67 89
Enter element to search 5
element not found
```

4. Write program in C to implement Insertion Sort

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

using namespace std;

int main() {

    int a[10], n;

    cout<<"enter size of array";

    cin>>n;

    cout<<"enter array elments ";

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

    {

        cin>>a[i];

    }

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

    { int current=a[i];

    int j=i-1;

    while(a[j]>current&& j>=0)

    {

    a[j+1]=a[j];

    j--;

    }

 a[j+1]=current;

        }

    cout<<"Sorted array is ";

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

    cout<<a[i]<<" ";

     return 0;

}
```

Output:

```
enter size of array5

enter array elments 4 7 98 34 65

Sorted array is 4 7 34 65 98
```

5. Write program in C to implement Merge Sort

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

using namespace std;

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

    int n1=m-l+1;

    int n2=r-m;

    int a[n1];

    int b[n2]; // temp arrays

    for(int i=0;i<n1;i++)

    {

        a[i]=arr[l+i];

    }

    for(int i=0;i<n2;i++)

    {

        b[i]=arr[m+1+i];

    }

    int i=0,j=0,k=l;

    while(i<n1 && j<n2)

    {

        if(a[i]<b[j])

        {

            arr[k]=a[i];
            k++;i++;

        }

        else {

            arr[k]=b[j];

            k++;j++;

        }

    }

    while(i<n1)

    {

        arr[k]=a[i];

            k++;i++;


    }

    while(j<n2)

    {

         arr[k]=b[j];

            k++;j++;

    }

}

void mergesort(int arr[],int l,int r)

{

    if(l<r)


{

  int mid=(l+r)/2;

  mergesort(arr,l,mid);

  mergesort(arr,mid+1,r);
  merge(arr,l,mid,r);
}
}
int main() {
     int a[10], n;
    cout<<"enter size of array ";
    cin>>n;
    cout<<"enter array elments ";
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
    }
   mergesort(a,0,n-1);
    cout<<"sorted arrray is ";
     for(int i=0;i<n;i++)
    {
        cout<<a[i]<<" ";
    }
    return 0;
}
```

Output:

```
enter size of array5
enter array elments 4 7 98 34 65
Sorted array is 4 7 34 65 98
```

6. Write program in C to implement Quick Sort

```
#include <bits/stdc++.h>
using namespace std;
int swap(int a[],int i,int j)
{
   int temp=a[i];
   a[i]=a[j];
   a[j]=temp;


}
int partition(int a[],int l,int r){
   int pivot=a[r];
   int i=l-1;
   for(int j=l;j<r;j++)
   {
      if(a[j]<pivot){
         i++;
         swap(a,i,j);
      }
   }
   swap(a,i+1,r);
   return i+1;
}
void quicksort(int a[],int l,int r)
{
   if(l<r){
      int pi=partition(a,l,r);
      quicksort(a,l,pi-1);
      quicksort(a,pi+1,r);
   }
}
int main() {
   int n;
   cout<<"enter size of array ";
   cin>>n;
   int a[n];

   cout<<"enter array elments ";
   for(int i=0;i<n;i++)
   {
      cin>>a[i];
   }
   quicksort(a,0,n-1);
   cout<<"sorted arrray is ";
    for(int i=0;i<n;i++)
   {
      cout<<a[i]<<" ";
   }
   return 0;
}
```

Output:

```
enter size of array5

enter array elments 4 7 98 34 65

Sorted array is 4 7 34 65 98
```

7. Write program in C to implement Selection Sort.

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

using namespace std;

int main() {

    int n;

    cout<<"enter size of array";cin>>n;

    int a[n];

 cout<<"enter array elments ";

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

    {

        cin>>a[i];

    }

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

        int smallest_idx = i;

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

            if(a[smallest_idx] > a[j]) {

                smallest_idx = j;

            }  }

        swap(a[smallest_idx], a[i]);

    }

    cout<<"Sorted array is ";

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

    {

        cout<<a[i]<<" ";

    }

    return 0;

}
```

Output:

```
enter size of array 5

enter array elments 4 7 98 34 65

Sorted array is 4 7 34 65 98
```

8. Write program in C to implement heap Sort

```
#include <bits/stdc++.h>
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 heapSort(int arr[], int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
```

```
        heapify(arr, n, i);
    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 n;
    cout<<"enter array size ";
    cin>>n;
    int arr[n];
    cout<<"enter array elements ";
    for (int i = 0; i < n; i++)
    {
        cin>>arr[i];
    }


    heapSort(arr, n);

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

```
Output:
enter array size 5
enter array elments 4 7 98 34 65
Sorted array is 4 7 34 65 98
```

9. Write algorithm and program in C to implement Counting Sort.

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

void countsort(int arr[],int n){
    int k=arr[0];
    for(int i=0;i<n;i++)
    {
        k=max(k,arr[i]);
    }
    int count[10]={0};
    for(int i=0;i<n;i++)
    {
        count[arr[i]]++;
    }
    for(int i=1;i<=k;i++)
    {
      count[i]+=count[i-1];
    }
    int output[n];
    for(int i=n-1;i>=0;i--){
        output[--count[arr[i]]]=arr[i];
    }
    for(int i=0;i<n;i++)
    {
        arr[i]=output[i];
    }

}
int main()
{

    int n;
    cout<<"enter array size ";
    cin>>n;
    int arr[n];
    cout<<"enter array elements ";
    for (int i = 0; i<n; i++)
    {
        cin >> arr[i];
    }
   countsort(arr, n);


    cout << "Sorted array is ";


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


    return 0;
}
```

```
Output:
enter array size 5

enter array elments 4 7 98 34 65

Sorted array is 4 7 34 65 98
```

10. Write algorithm and program in C to implement Radix Sort.

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

using namespace std;

void display(int *array, int size) {

   for(int i = 0; i<size; i++)

      cout << array[i] << " ";

   cout << endl;

}

void radixSort(int *arr, int n, int max) {
   int i, j, m, p = 1, index, temp, count = 0;
   list<int> pocket[10];
   for(i = 0; i< max; i++) {
      m = pow(10, i+1);
      p = pow(10, i);
      for(j = 0; j<n; j++) {
         temp = arr[j]%m;
         index = temp/p;
         pocket[index].push_back(arr[j]);
      }
      count = 0;
      for(j = 0; j<10; j++) {

         while(!pocket[j].empty()) {
            arr[count] = *(pocket[j].begin());
            pocket[j].erase(pocket[j].begin());
            count++;
         }
      }
   }
}
int main() {
   int n, max;
   cout << "Enter the number of elements: ";
   cin >> n;
   cout << "Enter the maximum digit of elements: ";
   cin >> max;
```

```
  int arr[n];
  cout << "Enter elements:" << endl;
  for(int i = 0; i<n; i++) {
    cin >> arr[i];
  }
  cout << "Data before Sorting: ";
  display(arr, n);
  radixSort(arr, n, max);
  cout << "Data after Sorting: ";
  display(arr, n);
}
```

Output:

```
Enter the number of elements: 5
Enter the maximum digit of elements: 3
Enter elements:
34 567 87 45 65
Data before Sorting: 34 567 87 45 65
Data after Sorting: 34 45 65 87 567
```

Assignment :2

1. WAP to implement Fractional KnapSack Problem

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

bool compare(pair<int, int> x, pair<int, int> y) {
        int f1 = x.second / x.first;
        int f2 = y.second / y.first;
        return f1 > f2;
}

int knapsack(vector<pair<int, int>> a, int W) {
        int ans = 0;
        sort(a.begin(), a.end(), compare);
        for(pair<int, int> i: a) {
                if(i.first <= W) {
                        ans += i.second;
                        W -= i.first;
                } else {
                        ans += (i.second / i.first) * W;
                        W = 0;
                        break;
                }
        }
        return ans;
}
```

```
int main() {
        int n, W;
   cout<<"enter number of items ";
   cin>>n;
   cout<<"enter capacity of knapsack ";
   cin>>W;

        vector<pair<int, int>> a;
   cout<<"enter weight and cost of items ";
   for(int i=0;i<n;i++){
                int f, s;
                cin>>f>>s;          // f --> weight , s --> cost
                a.push_back({f, s});
        }
        cout<< "Maximum profit= "<<knapsack(a, W);
        return 0;
}
```

Output:

```
enter number of items 5
enter capacity of knapsack 60
enter weight and cost of items 5 30
10 20
20 100
30 90
40 160
Maximum profit= 270
```

2. WAP to implement Integer KnapSack Problem

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

using namespace std;

#define lp(i,a,b) for(int i=a;i<b;i++)

const int N=1e3+2 ;

int val[N],wt[N];

int dp[N][N];

int knapsack(int n,int W)

{

   if(W<=0)

    return 0;

    if(n<=0)

    return 0;

    if(dp[n][W]!=-1)

    return dp[n][W];

    if(wt[n-1]>W)

    dp[n][W]= knapsack(n-1,W);

     else

    dp[n][W]= max(knapsack(n-1,W),knapsack(n-1,W-wt[n-1]) + val[n-1]);

    return dp[n][W];

    }

int main() {

    lp(i,0,N)

    {

        lp(j,0,N)

        dp[i][j]=-1;

    }
int n;
    cout<<" enter number of items ";
    cin>>n;
    int W;
    cout<<"enter weight of knapsack ";
    cin>>W;
   cout<<"enter weight of items ";
   lp(i,0,n)
   {
       cin>>wt[i];
   }
   cout<<"enter cost of items ";
   lp(i,0,n)
   {
       cin>>val[i];
       }
   cout<<"Maximum profit = "<<knapsack(n,W);
    return 0;
}
```

Output:

```
enter number of items 5
enter weight of knapsack 60
enter weight of items 5 10 20 30 40
enter cost of items 30 20 100 90 160
Maximum profit = 260
```

3. WAP to implement Matrix Chain Multiplication order

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

using namespace std;

int MatrixChainOrder(int p[], int i, int j)

{

   if (i == j)

      return 0;

   int k;

   int min = INT_MAX;

   int count;

   for (k = i; k < j; k++)

   {

      count = MatrixChainOrder(p, i, k)

            + MatrixChainOrder(p, k + 1, j)

            + p[i - 1] * p[k] * p[j];


      if (count < min)

         min = count;

   }


   return min;

}


int main()

{

    int n;

    cout<<"enter number of matrix ";
```

```
     cin>>n;

     int arr[n+1];

     cout<<"enter sequence ";

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

       cin>>arr[i];



    cout << "Minimum number of multiplications is "

         << MatrixChainOrder(arr, 1, n );

}
```

Output:

```
enter number of matrix 5

enter sequence  4 10 3 12 20 7

Minimum number of multiplications is 1344
```

4. WAP to implement Activity selector problem.

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

using namespace std;

int main() {

    int n;

    cout<<"enter number of activities";

    cin>>n;

    vector<vector<int>> v;

    cout<<"enter starting and ending time";

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

    {   int start,end;
```

```
        cin>>start>>end;
        v.push_back({start,end});
    }
    sort(v.begin(),v.end(),[&](vector<int>&a,vector<int>&b){
        return a[1]<b[1];
    });
    int take=1;
    int end=v[0][1];
    for(int i=1;i<n;i++)
    {
        if(v[i][0]>=end)
        {
            take++;
            end=v[i][1];
        }
    }
    cout<<"No of process  selected are ";
    cout<<take;
    return 0;
}
```

Output:

```
enter number of activities9
enter starting and ending time1 3
2 5
4 7
```

```
1 8

5 9

8 10

9 11

11 14

13 16

No of process selected   are 4
```

5. WAP to implement Longest Common Subsequence.

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

using namespace std;

int main() {


        string s1, s2;

   cout<<"enter first string ";

   cin >> s1;

   cout<<"enter second string ";

   cin>>s2;

        vector<vector<pair<int, string>>> dp;


        for(int i = s2.size() + 1; i >= 0; i--) {

                vector<pair<int, string>> a;

                for(int j = s1.size() + 1; j >= 0; j--) {

                        a.push_back({0, ""});

                }

                dp.push_back(a);

        }
```

```
    for(int i = s1.size() - 1; i >= 0; i--) {
        for(int j = s2.size() - 1; j >= 0;j--) {
            if(s1[i] == s2[j]) {
                dp[i][j].first = dp[i+1][j+1].first + 1;
                dp[i][j].second = s1[i] + dp[i+1][j+1].second;
            } else {
                string t1 = dp[i][j+1].second;
                string t2 = dp[i+1][j].second;
                if(t1.size() > t2.size()) {
                    dp[i][j] = {t1.size(), t1};
                } else {
                    dp[i][j] = {t2.size(), t2};
                }
            }
        }
    }

    cout<<"Size of LCS is"<<dp[0][0].first<<" and string is: "<<dp[0][0].second;
    return 0;
}
```

Output:

```
enter first string abcbdab

enter second string bdcaba

Size of LCS is4 and string is: bdab
```

Assignment:3

1,.WAP to implement BFS

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

using namespace std;


const int mod = 1e9+2, N = 1e5+2;


vector<int> graph[N];

vector<bool> vis(N, false);


int main() {


          int edges, vertices;

        cout<<"Enter the no of vertices and edges";

          cin>>vertices>>edges;

           cout<<"enter the edges";

          for(int i = 0; i<edges; i++) {

                    int x, y;cin>>x>>y;

                    graph[x].push_back(y);

                    graph[y].push_back(x);

          }

          queue<int> q;

          int source;cout<<"Enter the source: ";cin>>source;

          vis[source] = true;

          q.push(source);

          while(!q.empty()) {

                    int vertex = q.front();

                    q.pop();

                    cout<<vertex<<endl;

                    vector<int>::iterator it;
        for(it = graph[vertex].begin(); it != graph[vertex].end(); it++) {

            if(!vis[*it]) {

                vis[*it] = true;

                q.push(*it);

            }

        }

    }


    return 0;

}.
```

Output:

```
Enter the no of vertices and edges5 4

enter the edges0 1

0 2

2 3

3 4

Enter the source: 0

0

1

2

3

4
```

2. WAP to implement DFS.

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

#define vi vector<int>


const int N = 1e5+2, mod = 1e9+7;
vi graph[N];
vector<bool> vis(N, false);

void dfs(int source) {

    vis[source] = true;

    vector<int>::iterator it;
    for(it = graph[source].begin(); it != graph[source].end(); it++) {
        if(!vis[*it]) {
            dfs(*it);
        }
    }

    cout<<source<<endl;
}

int main() {
    int vertices, edges;
```

```
     cout<<"Enter the no of vertices and edges";
    cin>>vertices>>edges;
   cout<<"enter the edges"<<endl;
    for(int i = 0; i<edges; i++) {
        int x, y;cin>>x>>y;
        graph[x].push_back(y);
        graph[y].push_back(x);
    }

    vector<int> a;
    int source;cout<<"Enter the source: ";cin>>source;
    dfs(source);
    return 0;
}
```

Output:

```
Enter the no of vertices and edges5 4
enter the edges
0 1
0 2
2 3
3 4
Enter the source: 0
1
4
3
2
0
```

3.

WAP to implement Minimum spanning tree using Kruskal's Algorithm..

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

using namespace std;

const int N= 1e5+6;

vector<int> parent(N);

vector<int> sz(N);

void make_set(int V){

   parent[V]=V;

   sz[V]=1;

}

int find_set(int V){

   if(V==parent[V])

   return V;

return parent[V]=find_set(parent[V]);

}


void union_sets(int a,int b){

   a=find_set(a);

   b=find_set(b);

   if(a!=b){

      if(sz[a]<sz[b])

         swap(a,b);

      parent[b]=a;

      sz[a]+=sz[b];

   }
```

```
}

int main() {
    for(int i=0;i<N;i++)
    {
        make_set(i);
    }
    int n,m;
    cout<< "enter number of vertex ";
    cin>>n;
    cout<< "enter number of edges ";
    cin>>m;

    vector<vector<int>> edges;
    cout<<"enter edges vertex and weight ";
    for(int i=0;i<m;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        edges.push_back({w,u,v});
    }
    sort(edges.begin(),edges.end());
     int cost=0;
     for(auto i:edges){
         int w=i[0];
         int u=i[1];
        int V=i[2];
        int x=find_set(u);
        int y=find_set(V);

        if(x==y)
            continue;
         else
         {
             cost+=w;
             union_sets(u,V);
         }
     }
     cout<<"Cost of minimum spanning tree is "<<cost;
    return 0;
}
```

Output:

```
enter number of vertex 7
enter number of edges 9
enter edges vertex and weight 0 5 10
0 1 28
5 4 25
1 6 14
1 2 16
3 6 24
4 3 22
6 3 18
3 2 12
Cost of minimum spanning tree is 99
```

4. WAP to implement Minimum spanning tree using Prim's Algorithm

```
#include <bits/stdc++.h>
using namespace std;
const int mod = 1e9+7, N = 1e5+2;
int main() {
        int vertices, edges;
        cout<<"Enter the no of vertices and edges";
         cin>>vertices>>edges;
        vector<pair<int, int>> adj[N];
        for(int i = 0; i<edges; i++) {
                int x, y, wt;
                cin>>wt>>x>>y;
                adj[x].push_back({y, wt});
                adj[y].push_back({x, wt});
        }

        vector<int> parent(N);
        vector<int> key(N);
        vector<bool> mstSet(N);

        for(int i = 0; i<vertices; i++) {
                key[i] = INT_MAX;
                mstSet[i] = false;
                parent[i] = -1;
        }

        key[0] = 0;
}

parent[0] = -1;

for(int count = 0; count < vertices - 1; count++) {
    int mi = INT_MAX, u;

    for(int v = 0; v < vertices; v++) {
        if(!mstSet[v] and key[v] < mi) {
            mi = key[v];
            u = v;
        }
    }

    mstSet[u] = true;

    for(auto i: adj[u]) {
        int v = i.first;
        int weight = i.second;
        if(!mstSet[v] and weight < key[v]) {
            parent[v] = u;
            key[v] = weight;

        }
    }
}
```

```
cout<<”MST is :”

for(int i = 0; i < vertices; i++) {

                        cout<<parent[i] << " - "<<i<<endl;

            }

            return 0;

}
```

Output:

```
Enter the no of vertices and edges7 9

10 0 5

28 0 1

25 5 4

24 4 6

14 1 6

22 4 3

18 6 3

16 1 2

12 3 2

MST is :

-1 - 0

2 - 1

3 - 2

4 - 3

5 - 4

0 - 5

1 - 6
```

5.

WAP to implement single source shortest path using Dikjastra's Algorithm.

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

using namespace std;

#define ll long long

#define vi vector<int>


const int mod = 1e9+7, inf = 1e5+2;

int main() {

    int vertices, edges;

    cout<<"Enter the no of vertices and edges";

   cin>>vertices>>edges;

    vi dist(vertices+1, inf);

    vector<vector<pair<int, int>>> graph(vertices+1);

    cout<<"Enter the  edges with weight :";

*

    for(int i = 0; i<edges; i++) {

        int x, y, w;

        cin>>x>>y>>w;

        graph[x].push_back({w, y});

        graph[y].push_back({w, x}); // first-> weight, second -> vertex

    }

    int source;

    cout<<"Enter the source: ";cin>>source;

    dist[source] = 0;
    set<pair<int, int>> s;
    s.insert({0, source}); // first -> distance, second -> vertex
    while (!s.empty())
    {
        auto u = *s.begin();
        s.erase(u);
        for(auto i: graph[u.second]) {
            if(dist[i.second] > dist[u.second] + i.first) {
                s.erase({dist[i.second], i.second});
                dist[i.second] = dist[u.second] + i.first;
                s.insert({dist[i.second], i.second});
            }
        }
    }
    cout<<"Distance between "<<source<<" and vertex : "<<endl;

    for(int i = 0; i<=vertices; i++) {
        if(dist[i] < inf) cout<<i<<": "<<dist[i]<<endl;
        else cout<<i<<": -1"<<endl;
    }

    return 0;
}
```

Output:

```
enter no of vertices and edges  5 9

enter the edegs with weight

0 1 10

0 2 3

1 2 1

2 1 4

2 3 8

1 3 2

3 4 7

4 3 9

2 4 2

Enter the source: 0

Distance between0 and vertex:

2: 3

1: 7

3: 9

4:5
```