**Q1\_selection.cpp**

#include<iostream>

using namespace std;

int bsearch(int \*arr,int s,int e,int k)

{

if(s>e){

return -1;

}

int mid=(s+e)/2;

if(arr[mid]==k){

return mid;

}

else if(arr[mid]>k){

bsearch(arr,s,mid-1,k);

}

else{

bsearch(arr,mid+1,e,k);

}

}

void ssort(int \*arr,int n)

{

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

int mi=i;

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

if(arr[j]<arr[mi]){

mi=j;

}

}

swap(arr[i],arr[mi]);

}

}

int main()

{

int n;

cout<<"enter the length ";

cin>>n;

cout<<"enter the elements "<<endl;

int \*arr= new int[n];

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

cin>>arr[i];

}

ssort(arr,n);

cout<<"after sorting ";

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

cout<<arr[i]<<" ";

}

cout<<endl;

int k;

cout<<"enter the element for searching";

cin>>k;

cout<<bsearch(arr,0,n,k);

}

**Q2\_merge\_sort.cpp**

#include<iostream>

using namespace std;

void merge(int arr[], int left, int mid, int right){

int nl, nr;

nl= mid - left +1;

nr= right - mid;

int L[nl], R[nr];

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

L[i]=arr[left+i];

for(int j=0; j<nr; j++)

R[j]=arr[mid+1+j];

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

while(i<nl && j<nr){

if(L[i] <= R[j]){

arr[k]=L[i];

i++; k++;

}

else{

arr[k]=R[j];

j++; k++;

}

// k++;

}

while(i < nl){

arr[k] = L[i];

i++; k++;

}

while(j < nr){

arr[k] = R[j];

j++; k++;

}

}

void mergeSort(int arr[], int left, int right){

int mid;

if(right > left){

mid = (left + right)/2;

mergeSort(arr, left, mid);

mergeSort(arr, mid+1, right);

merge(arr, left, mid, right);

}

}

int main(){

int n;

cout<<"Enter size of an array: ";

cin>>n;

int arr[n];

cout<<"Enter elements: ";

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

cin>>arr[i];

cout<<"Before sorting array: ";

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

cout<<arr[i]<<" ";

cout<<endl;

mergeSort(arr, 0, n-1);

cout<<"After sorting array: ";

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

cout<<arr[i]<<" ";

}

Q3\_Quick\_sort.cpp

#include<iostream>

using namespace std;

int partition(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(int arr[], int low, int high){

if(low< high){

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

quickSort(arr,low,pi-1);

quickSort(arr,pi,high);

}

}

int main(){

int n;

cout<<"Enter size of an array: ";

cin>>n;

int arr[n];

cout<<"Enter elements: ";

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

cin>>arr[i];

cout<<"Before sorting array: ";

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

cout<<arr[i]<<" ";

cout<<endl;

quickSort(arr, 0, n-1);

cout<<"After sorting array: ";

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

cout<<arr[i]<<" ";

}

**Q4\_Heap\_sort.cpp**

#include<iostream>

using namespace std;

void heapify (int arr[], int n, int i){

int len=i;

int left=2\*i + 1;

int right = 2\*i + 2;

if(left<n && arr[left]>arr[len])

len=left;

if(right<n && arr[right]>arr[len])

len=right;

if(len!=i){

swap(arr[i],arr[len]);

heapify(arr,n,len);

}

}

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

for(int i=i/2; i>=0; i--)

heapify(arr,n,i);

}

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

buildMaxHeap(arr,n);

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

swap(arr[i],arr[0]);

heapify(arr,i,0);

}

}

int main(){

int n;

cout<<"Enter size of an array: ";

cin>>n;

int arr[n];

cout<<"Enter elements: ";

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

cin>>arr[i];

cout<<"Before sorting array: ";

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

cout<<arr[i]<<" ";

cout<<endl;

heapSort(arr, n);

cout<<"After sorting array: ";

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

cout<<arr[i]<<" ";

}

Q5\_Knap\_sack.c

#include <stdio.h>

void main()

{

int capacity, no\_items, cur\_weight, item;

int used[10];

float total\_profit;

int i;

int weight[10];

int value[10];

printf("Enter the capacity of knapsack:\n");

scanf("%d", &capacity);

printf("Enter the number of items:\n");

scanf("%d", &no\_items);

printf("Enter the weight and value of %d item:\n", no\_items);

for (i = 0; i < no\_items; i++)

{

printf("Weight[%d]:\t", i);

scanf("%d", &weight[i]);

printf("Value[%d]:\t", i);

scanf("%d", &value[i]);

}

for (i = 0; i < no\_items; ++i)

used[i] = 0;

cur\_weight = capacity;

while (cur\_weight > 0)

{

item = -1;

for (i = 0; i < no\_items; ++i)

if ((used[i] == 0) &&

((item == -1) || ((float) value[i] / weight[i] > (float) value[item] / weight[item])))

item = i;

used[item] = 1;

cur\_weight -= weight[item];

total\_profit += value[item];

if (cur\_weight >= 0)

printf("Added object %d (%d Rs., %dKg) completely in the bag. Space left: %d.\n", item + 1, value[item], weight[item], cur\_weight);

else

{

int item\_percent = (int) ((1 + (float) cur\_weight / weight[item]) \* 100);

printf("Added %d%% (%d Rs., %dKg) of object %d in the bag.\n", item\_percent, value[item], weight[item], item + 1);

total\_profit -= value[item];

total\_profit += (1 + (float)cur\_weight / weight[item]) \* value[item];

}

}

printf("Filled the bag with objects worth %.2f Rs.\n", total\_profit);

}

Q 6\_floydwarshal.cpp

#include<bits/stdc++.h>

using namespace std;

#define N 5

#define inf 999

int w[N][N]=

{{0,3,8,inf,-4},

{inf,0,inf,1,7},

{inf,4,0,inf,inf},

{2,inf,-5,0,inf},

{inf,inf,inf,6,0}

};

int main(){

for(int k=0; k<N; k++){

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

for(int j=0; j<N; j++){

if(w[i][k]+w[k][j] < w[i][j])

w[i][j]=w[i][k]+w[k][j];

}

}

cout<<"D("<<k<<"):"<<endl;

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

for(int j=0; j<N; j++){

if(w[i][j]==inf)

cout<<setw(5)<<"INF";

else

cout<<setw(5)<<w[i][j];

}

cout<<endl;

}

cout<<endl;

}

    return 0;

}

Q 7\_Matrix\_chain.c

#include <stdio.h>

#include<limits.h>

#define INFY 999999999

long int m[20][20];

int s[20][20];

int p[20],i,j,n;

void matmultiply(void)

{

long int q;

int k;

for(i=n;i>0;i--)

{

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

{

if(i==j)

m[i][j]=0;

else

{

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

{

q=m[i][k]+m[k+1][j]+p[i-1]\*p[k]\*p[j];

if(q<m[i][j])

{

m[i][j]=q;

s[i][j]=k;

}

}

}

}

}

}

int main()

{

int k;

printf("Enter the no. of elements: ");

scanf("%d",&n);

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

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

{

m[i][i]=0;

m[i][j]=INFY;

s[i][j]=0;

}

printf("\nEnter the dimensions: \n");

for(k=0;k<=n;k++)

{

printf("P%d: ",k);

scanf("%d",&p[k]);

}

matmultiply();

printf("\nCost Matrix M:\n");

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

{

for(j=1;j<=n;j++)

{

if(i>j)

printf("\t");

else

printf("%ld\t",m[i][j]);

}

printf("\n");

}

printf("\nPartition Matrix M:\n");

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

{

for(j=1;j<=n;j++)

{

if(i>=j)

printf("\t");

else

printf("%d\t",s[i][j]);

}

printf("\n");

}

printf("Enter value of i:");

scanf("%d",&i);

printf("Enter value of j:");

scanf("%d",&j);

printf("\nMultiplication Sequence : ");

printf("\nMinimum cost is : %d ",m[i][j]);

printf("\nValue of k for partition is : %d ",s[i][j]);

return 0;

}

Q Bubble\_sort.c

#include<stdio.h>

void swaps(int \*x,int \*y){

int temp =\*x;

\*x=\*y;

\*y=temp;

}

void bubblesort(int a[],int n){

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

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

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

swaps(&a[j],&a[j+1]);

}

}

}

}

int main(){

int a[50],n,i;

printf("enter the size of array:\n");

scanf("%d",&n);

printf("enter the elements of array:\n");

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

scanf("%d",&a[i]);

}

printf("element before sorting;\n");

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

printf("%d\t",a[i]);

}

bubblesort(a,n);

printf("\narray after sorting \n");

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

printf("%d\t",a[i]);

}

     return 0;

}