

INSERTION SORT

```
1  #include<stdio.h>
2  void insertion(int arr[],int n){
3      for(int i=1;i<n;i++){
4          int key=arr[i];
5          int j=i-1;
6          while((j>=0) && (arr[j]>key)){
7              arr[j+1]=arr[j];
8              j--;
9          }
10         arr[j+1]=key;
11     }
12 }
13 int main(){
14     int n;
15     printf("Enter size of the array: ");
16     scanf("%d",&n);
17     int arr[n];
18     printf("Enter %d elements in to the array: ",n);
19     for(int i=0;i<n;i++)
20         scanf("%d",&arr[i]);
21     insertion(arr,n);
22     printf("After sorting the elements are:");
23     for(int i=0;i<n;i++)
24         printf(" %d",arr[i]);
25     return 0;
26 }
```

BINARY SEARCH

```
1  #include<stdio.h>
2  int binary(int arr[],int k, int low,int high){
3      if(low<=high){
4          int mid=(low+high)/2;
5          if(arr[mid]==k)
6              return mid;
7          else if(arr[mid]<k)
8              return binary(arr,k,mid+1,high);
9          return binary(arr,k,low,mid-1);
10     }
11     return -1;
12 }
13 int main(){
14     int n,k;
15     printf("Enter number of elements: ");
16     scanf("%d",&n);
17     printf("Enter the sorted array: ");
18     int arr[n];
19     for(int i=0;i<n;i++)
20         scanf("%d",&arr[i]);
21     printf("enter the item to be search: ");
22     scanf("%d",&k);
23     int a = binary(arr,k,0,n-1);
24     if(a== -1)
25         printf("item not present");
26     else
27         printf("item present");
28     return 0;
29 }
```

COUNT SORT

```
1    #include<stdio.h>
2
3    void countSort(int arr[],int n){
4        int k=arr[0];
5        for(int i=0;i<n;i++){
6            k=k<arr[i]?arr[i]:k;
7            int count[123]={0};
8            for(int i=0;i<n;i++){
9                count[arr[i]]++;
10           }
11           for(int i=1;i<=k;i++){
12               count[i]+=count[i-1];
13           }
14           int output[n];
15           for(int i=n-1;i>=0;i--){
16               output[--count[arr[i]]]=arr[i];
17           }
18           for(int i=0;i<n;i++){
19               arr[i]=output[i];
20           }
21       }
22
23   int main(){
24       int n;
25       printf("enter the no. of array element: ");
26       scanf("%d",&n);
27       int arr[n];
28       printf("enter the element: ");
29       for(int i=0;i<n;i++){
30           scanf("%d",&arr[i]);
31           countSort(arr,n);
32       }
33       for(int i=0;i<n;i++){
34           printf("%d ",arr[i]);
35       }
36       return 0;
37   }
```

LINEAR SEARCH

```
1  #include<stdio.h>
2  int recursive(int a[],int item,int p,int r){
3      if(r<p)
4          return 1;
5      else if(a[p]==item)
6          return p;
7      else
8          recursive(a,item,p+1,r);
9  }
10 int main(){
11     int val,i,item,p,r,n,m;
12     printf("enter the no of elements: ");
13     scanf("%d",&n);
14     int a[n];
15     printf("Enter %d integer(s)\n",n);
16     for(i=0;i<n;i++)
17         scanf("%d",&a[i]);
18     printf("enter the item to be search: ");
19     scanf("%d",&m);
20     p=0,r=n;
21     val=recursive(a,m,p,r);
22     if(val==1)
23         printf("no item found");
24     else
25         printf("item location = %d  item = %d",val+1,m);
26     return 0;
27 }
```

SELECTION SORT

```
1  #include<stdio.h>
2
3  void selection(int arr[],int n){
4
5      for(int i=0;i<n;i++)
6
7          {
8
9              for(int j=i+1;j<n;j++){
10
11                  if(arr[i]>arr[j]){
12
13                      arr[i]=arr[i]^arr[j];
14
15                      arr[j]=arr[i]^arr[j];
16
17                      arr[i]=arr[i]^arr[j];
18
19                  }
20
21              }
22
23          }
24
25  }
26
27  int main(){
28
29      int n;
30
31      // scanf("%d",&n)
32
33      printf("Enter size of the array : ");
```

```
34
35     scanf("%d",&n);
36
37     int arr[n];
38
39     printf("Enter the elements :");
40
41     for(int i=0;i<n;i++)
42
43         scanf("%d",&arr[i]);
44
45     selection(arr,n);
46
47     printf("The sorted elements are : ");
48
49     for(int i=0;i<n;i++)
50
51         printf("%d\t",arr[i]);
52
53     return 0;
54
55 }
```

MERGE SORT

```
1  #include<stdio.h>
2  #include<conio.h>
3  void merge(int arr[],int l,int m,int r){
4      int i,j,k;
5      int n1=m-l+1;
6      int n2=r-m;
7      int L[n1],R[n2];
8      for(i=0;i<n1;i++)
9          L[i]=arr[l+i];
10     for(j=0;j<n2;j++)
11         R[j]=arr[m+1+j];
12     i=0;
13     j=0;
14     k=l;
15     while(i<n1 && j<n2){
16         if(L[i]<R[j]){
17             arr[k]=L[i];
18             i++;
19         }
20         else{
21             arr[k]=R[j];
22             j++;
23         }
24         k++;
25     }
26     while(i<n1){
27         arr[k]=L[i];
28         i++;
29         k++;
30     }
31     while(j<n2){
32         arr[k]=R[j];
33         j++;
34         k++;
35     }
36 }
37 void merge_sort(int arr[],int l,int r)
38 {
39     if(l<r){
40         int m=(l+r)/2;
41         merge_sort(arr,l,m);
42         merge_sort(arr,m+1,r);
43         merge(arr,l,m,r);
44     }
45 }
46 void main(){
47     int n,i,a[100];
48     printf(" Enter How many Numbers : ");
49     scanf("%d",&n);
50     printf(" Enter %d Numbers :",n);
51     for(i=0;i<n;i++){
52
53         scanf("%d",&a[i]);
54
55     }
56     merge_sort(a,0,n-1);
57     printf(" Sorted Numbers are : ");
58     for(i=0;i<n;i++){
59         {
60
61             printf("%d\t",a[i]); //use tab instead of space here
62
63         }
64     }
```

QUICK SORT

```
1  #include <stdio.h>
2  void quicksort(int [], int, int);
3  int main(){
4      int list[50];
5      int size,i;
6      printf("Enter Number of elements : ");
7      scanf("%d", &size);
8      printf("Enter %d Elements : ",size);
9      for(i=0;i<size;i++){
10         scanf("%d",&list[i]);
11     }
12     quicksort(list,0,size-1);
13     printf("Sorted Numbers are :");
14     for(i=0; i<size; i++){
15         printf(" %d", list[i]);
16     }
17     printf(" \n");
18     return 0;
19 }
20 void quicksort(int list[], int low, int high){
21     int pivot, i, j, temp;
22     if(low<high){
23         pivot=low;
24         i=low;
25         j=high;
26         while(i<j){
27             while(list[i]<=list[pivot]&& i<=high){
28                 i++;
29             }
30             while(list[j]>list[pivot]&&j>=low){
31                 j--;
32             }
33             if(i<j){
34                 temp=list[i];
35                 list[i]=list[j];
36                 list[j]=temp;
37             }
38         }
39         temp=list[j];
40         list[j]=list[pivot];
41         list[pivot]=temp;
42         quicksort(list,low,j-1);
43         quicksort(list,j+1,high);
44     }
45 }
```


DIVIDE AND CONQUER

```
1  #include<stdio.h>
2  #include<conio.h>
3  void main(){
4      int arr[10],n,i,max,min;
5      printf("Enter the total number of Elements : ");
6      scanf("%d",&n);
7      printf("Enter the numbers : ");
8      for(i=0;i<n;i++){
9          scanf("%d",&arr[i]);
10     }
11     max=min=arr[0];
12     for(i=0;i<n;i++){
13         if(arr[i]>max){
14             max=arr[i];
15         }
16         if(arr[i]<min){
17             min=arr[i];
18         }
19     }
20     printf("Minimum element in an array : %d",min);
21     printf("\nMaximum element in an array : %d\n",max);
22 }
```

Heap Sort

```
1  #include<stdio.h>
2  #include<conio.h>
3  int temp;
4  void heap(int arr[10],int n,int i){
5      int largest=i;
6      int left=2*i+1;
7      int right=2*i+2;
8      if(left<n&& arr[left]>arr[largest])
9          largest=left;
10     if(right<n && arr[right]>arr[largest])
11         largest=right;
12     if(largest!=i){
13         temp=arr[i];
14         arr[i]=arr[largest];
15         arr[largest]=temp;
16         heap(arr,n,largest);
17     }
18 }
19 void heapsort(int arr[],int n){
20     int i;
21     for(i=n/2-1;i>=0;i--){
22         heap(arr,n,i);
23     }
24     for(i=n-1;i>=0;i--){
25         temp=arr[0];
26         arr[0]=arr[i];
27         arr[i]=temp;
28         heap(arr,i,0);
29     }
30 }
31 void main(){
32     int i,n,a[10];
33     printf("enter the no. of element: ");
34     scanf("%d",&n);
35     printf("Enter elements: ");
36     for(i=0;i<n;i++){
37         scanf("%d",&a[i]);
38     }
39     heapsort(a,n);
40     for(i=0;i<n;i++){
41         printf("%d\t",a[i]);
42     }
```

FLOYD

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<limits.h>
4  int p[20][20];
5  int d[20][20];
6  int w[20][20];
7  void print_path(int i,int j){
8      if(i==j)
9          printf("%d",i);
10     else{
11         if(p[i][j]==-1)
12             printf("No path Exists");
13         else{
14             print_path(i,p[i][j]);
15             printf("-> %d",j);
16         }
17     }
18 }
19 void warshall(int n){
20     for(int i=1;i<=n;i++){
21         for(int j=1;j<=n;j++){
22             d[i][j]=w[i][j];
23         }
24     }
25     for(int k=1;k<=n;k++){
26         for(int i=1;i<=n;i++){
27             for(int j=1;j<=n;j++){
28                 if(d[i][k]==INT_MAX || d[k][j]==INT_MAX)
29                     continue;
30                 if(d[i][k]+d[k][j]<d[i][j])
31                 {
32                     d[i][j]=d[i][k]+d[k][j];
33                     p[i][j]=p[k][j];
34                 }
35             }
36         }
37     }
```

```

38     }
39     void main(){
40         int i,j,v,s,des;
41         char ch;
42         printf("Enter number of vertices: ");
43         scanf("%d",&v);
44         printf("Enter the weight matrix");
45         for(i=1;i<=v;i++)
46         {
47             for(j=1;j<=v;j++)
48             {
49                 if(i==j)
50                 {
51                     w[i][j]=0;
52                     p[i][j]=-1;
53                     continue;
54                 }
55                 printf("Is edge (%d,%d) present in graph (y/n): ",i,j);
56                 fflush(stdin);
57                 scanf("%c",&ch);
58                 if(ch=='y' || ch=='Y'){
59                     printf("Enter weight of edge (%d,%d): ",i,j);
60                     scanf("%d",&w[i][j]);
61                     p[i][j]=i;
62                 }
63                 else{
64                     w[i][j]=INT_MAX;
65                     p[i][j]=-1;
66                 }
67             }
68         }
69         warshall(v);
70         printf("Enter source and destination: ");
71         scanf("%d %d",&s,&des);
72         printf("Distance = %d",d[s][des]);
73         print_path(s,des);
74     }

```

KNAPSACK

```
1  #include<stdio.h>
2  #include<conio.h>
3  int max(int a, int b){
4      return(a>b)?a:b;
5  }
6  int knapsack(int W,int v[],int w[],int n){
7      if(n==0||W==0)
8          return 0;
9      if(w[n-1]>W)
10         return knapsack(W,v,w,n-1);
11     else
12         return max(v[n-1]+knapsack(W-w[n-1],v,w,n-1),knapsack(W,v,w,n-1));
13 }
14 void main(){
15     int n,W;
16     printf("Enter number of items:");
17     scanf("%d",&n);
18     int v[n],w[n];
19     printf("Enter value and weight of items:");
20     for(int i=0;i<n;i++){
21         scanf("%d %d",&v[i],&w[i]);
22     }
23     printf("Enter size of knapsack:");
24     scanf("%d",&W);
25     printf("Maximum value in 0/1 knapsack :%d",knapsack(W,v,w,n));
26 }
```

CHAIN MULTIPLICATION

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<limits.h>
4  int m[20][20],s[20][20];void Print_optimal_parens(i,j){
5      if(i==j)
6      {
7          printf("A%d",i);
8      }
9      }
10     else
11     {
12         printf("(");
13         Print_optimal_parens(i,s[i][j]);
14         Print_optimal_parens(s[i][j]+1,j);
15         printf(")");
16     }
17 }
18
19 }
20 void Matrix_chain_order(int p[],int n){
21     int q,j,i,l,k;
22     for(i=1;i<=n;i++)
23     {
24         m[i][i]=0;
25     }
26 }
27 for(l=2;l<=n;l++)
28 {
29     for(i=1;i<=n-l+1;i++)
30     {
```

```

31             j=i+1-1;
32             m[i][j]=INT_MAX;
33             for(k=i;k<=j-1;k++)
34             {
35                 q=m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
36                 if(q<m[i][j])
37                 {
38                     m[i][j]=q;
39                     s[i][j]=k;
40                 }
41             }
42         }
43     }
44 }
45
46
47 }
48 Print_optimal_parens(1,n);
49
50 }
51 void main(){
52     int n;
53     printf("enter the matrices");
54     scanf("%d",&n);
55     int p[n];
56     for(int i=0;i<=n;i++)
57     {
58         scanf("%d",&p[i]);
59     }
60 }
61 Matrix_chain_order(p,n);
62 printf("%d",m[1][n]);
63 }

```

LARGEST SUBSEQUENCE

```
1  #include<stdio.h>
2  #include<conio.h>
3  void lcs(char a[],char b[]){
4      int n=strlen(a);
5      int m=strlen(b);
6      int c[n+1][m+1];
7      for(int j=0;j<=m;j++){
8          c[0][j]=0;
9      }
10     for(int i=1;i<=n;i++){
11         c[i][0]=0;
12     }
13     for(int i=1;i<=n;i++){
14         for(int j=1;j<=m;j++){
15             {
16                 if(a[i-1]==b[j-1])
17                     c[i][j]=c[i-1][j-1]+1;
18                 else if(c[i-1][j]>=c[i][j-1])
19                     c[i][j]=c[i-1][j];
20                 else
21                     c[i][j]=c[i][j-1];
22             }
23         }
24         printf("Length of LCS is %d\n",c[n][m]);
25     }
26     void main(){
27         char a[50],b[50];
28         printf("Enter a string1: ");
29         gets(a);
30         printf("Enter a string2: ");
31         gets(b);
32         lcs(a,b);
33     }
```

N QUEEN PROBLEM

```
1  #include<stdio.h>
2  #include<conio.h>
3  int board[20],count;
4  int main(){
5      int n,i,j;
6      void queen(int row,int n);
7      printf("Enter number of Queens: ");
8      scanf("%d",&n);
9      queen(1,n);
10     return 0;
11 }
12
13 void print(int n){
14     int i,j;
15     for(i=1;i<=n;i++){
16         for(j=1;j<=n;j++){
17             if(board[i]==j){
18                 printf("row no %d\tcolom no %d\n",i,j);
19             }
20         }
21     }
22 }
23 int place(int row,int column){
24     int i;
25     for(i=1;i<=row-1;++i){
26         if(board[i]==column){
27             return 0;
28         }
29         else if(abs(board[i]-column)==abs(i-row)){
30             return 0;
31         }
32     }
33     return 1;
34 }
35 void queen(int row, int n){
36     int column;
37     for(column=1;column<=n;++column){
38         if(place(row,column)){
39             board[row]=column;
40             if(row==n)
41                 print(n);
42             else
43                 queen(row+1,n);
44         }
45     }
46 }
```

PRIMS

```
1  #include <stdio.h>
2  int a,b,u,v,n,i,j,ne=1;
3  int visited[10]= { 0 },min,mincost=0,cost[10][10];
4  int main(){
5      printf("To compute the spanning tree from the adjacency matrix");
6      printf("\nHow many nodes :");
7      scanf("%d",&n);
8      printf("Enter the adjacency matrix :");
9      for (i=1;i<=n;i++)
10         for (j=1;j<=n;j++){
11             scanf("%d",&cost[i][j]);
12             if(cost[i][j]==0)
13                 cost[i][j]=999;
14
15         }
16     printf("The entered adjacency matrix :\n");
17     for(i=1;i<=n;i++){
18         for(j=1;j<=n;j++){
19             if(cost[i][j]==999)
20                 printf("%-3d",0);
21             else
22                 printf("%-3d",cost[i][j]);
23         }
24         printf("\n");
25     }
26     visited[1]=1;
27     printf("The nodes to be connected in spanning tree are : ");
28     while(ne<n){
29         for (i=1,min=999;i<=n;i++)
30             for (j=1;j<=n;j++){
31                 if(cost[i][j]<min)
32                     if(visited[i]!=0)
33                     {
34                         min=cost[i][j];
35                         a=u=i;
36                         b=v=j;
37
38                     }
39                 if(visited[u]==0 || visited[v]==0)
40                 {
41                     printf("(%d,%d);",a,b);
42                     ne++;
43                     mincost+=min;
44                     visited[b]=1;
45
46                 }
47                 cost[a][b]=cost[b][a]=999;
48
49             }
50     printf("\nThe cost of Minimum Spanning Tree is :%d",mincost);
51     return 0;
52 }
```

KRUSKAL

```
1  #include<conio.h>
2  int parent[100];
3  int find(int i)
4  {
5      while(parent[i]!=i)
6          i=parent[i];
7      return i;
8  }
9  void unio(int i,int j){
10     int x,y;
11     x=find(i);
12     y=find(j);
13     parent[x]=y;
14 }
15 void kruskal(int a[][100],int n){
16     int k,co=0,min,r,b,l,res[100][2];
17     for(k=0;k<n;k++)
18         parent[k]=k;
19     printf("The minimum spanning tree has the following edges:\n");
20     while(co<n-1){
21         min=10000000;
22         r=-1;
23         b=-1;
24         for(k=n-1;k>-1;k--){
25             for(l=n-1;l>-1;l--){
26                 if(find(k)!=find(l) && a[k][l]<min && a[k][l]!=0){
27                     min=a[k][l];
28                     r=k;
29                     b=l;
30                 }
31             }
32         }
33         unio(r,b);
34         res[co][0]=r+1;
35         res[co][1]=b+1;
```

```

36         co++;
37     }
38     for(k=n-2;k>-1;k--)
39         printf("%d-%d\n",res[k][0],res[k][1]);
40 }
41 void main(){
42     char c;
43     int n,i,j,a[100][100],l[1000];
44     printf("Input as adjacency matrix or adjacency list?(A/E)");
45     scanf("%c",&c);
46     printf("no of nodes :");
47     scanf("%d",&n);
48     printf("Input as adjacency matrix:\n");
49     for(i=0;i<n;i++){
50         printf("Row %d:",i+1);
51         for(j=0;j<n;j++){
52             scanf("%d",&a[i][j]);
53         }
54     }
55     kruskal(a,n);
56 }

```

KNAPSACK

```
#include<stdio.h>
```

```
void knapsack(int n,float weight[], float profit[],float capacity){
```

```
    float x[20],tp=0;
```

```
    int i,j,u;
```

```
    u=capacity;
```

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

```
        x[i]=0.0;
```

```
    }
```

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

```
        if(weight[i]>u)
```

```
            break;
```

```
        else{
```

```
            x[i]=1.0;
```

```
            tp=tp+profit[i];
```

```
            u=u-weight[i];
```

```
        }
```

```
    }
```

```
    if(i<=n){
```

```
        x[i]=u/weight[i];
```

```
    }
```

```
    tp=tp+(x[i]*profit[i]);
```

```
    printf("The result vector is:- \n");
```

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

```
        printf("%.2f\t",x[i]);
```

```
    printf("\nMaximum profit is:- %.2f",tp);
```

```
}
```

```
int main(){
```

```
    float weight[20],profit[20],capacity;
```

```
    int num,i,j;
```

```
    float ratio[20],temp;
```

```
    printf("Enter the no. of objects:- ");
```

```

scanf("%d", &num);

printf("Enter the Weight, Value(Profit) of each object:- \n");

for(i=1;i<=num;i++){
    printf("item %d:",i);
    scanf("%f%f",&weight[i],&profit[i]);
}

printf("Enter the capacity of knapsack:- ");
scanf("%f",&capacity);

for(i=1;i<=num;i++){
    ratio[i]=profit[i]/weight[i];
}

for(i=1;i<=num;i++){
    for(j=i+1;j<=num;j++){
        if(ratio[i]<ratio[j]){
            temp=ratio[j];
            ratio[j]=ratio[i];
            ratio[i]=temp;
            temp=weight[j];
            weight[j]=weight[i];
            weight[i]=temp;
            temp=profit[j];
            profit[j]=profit[i];
            profit[i]=temp;
        }
    }
}

knapsack(num,weight,profit,capacity);

return (0);
}

```