

GCD RECURSIVE

```
1  #include<stdio.h>
2  int gcd(int x,int y);
3  int main()
4  {
5      int x,y;
6      printf("Enter The First Number\n");
7      scanf("%d",&x);
8      printf("Enter the Second Number\n");
9      scanf("%d",&y);
10     int ans=gcd(x,y);
11     printf("GCD OF %d AND %d is %d",x,y,ans);
12
13 }
14 int gcd(int x,int y)
15 {
16     int rem;
17     int temp;
18     if(x>y)
19     {
20         rem=x%y;
21     }
22     else
23     {
24         temp=y;
25         y=x;
26         x=temp;
27         rem=x%y;
28     }
29     if(rem==0)
30     {
31         return y;
32     }
33     else
34     {
35         return gcd(y,rem);
36     }
37 }
```

Enter The First Number

25

Enter the Second Number

5

GCD OF 25 AND 5 is 5

Process returned 0 (0x0) execution time : 11.009 s

Press any key to continue.

GCD ITERATIVE

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<ctype.h>
4  int gcd(int x,int y);
5  int main()
6  {
7      int x,y;
8      printf("Enter The First Number \n");
9      scanf("%d",&x);
10     printf("Enter the Second Number\n");
11     scanf("%d",&y);
12     int ans=gcd(x,y);
13     printf("GCD OF %d AND %d is %d",x,y,ans);
14 }
15 int gcd(int x,int y)
16 {
17     int rem,temp;
18     if(x>y)
19     {
20         rem=x%y;
21     }
22     else
23     {
24         temp=x;
25         x=y;
26         y=temp;
27         rem=x%y;
28     }
29     if(rem==0)
30     {
31         return y;
32     }
33     else
34     {
35         while(rem!=0)
36         {
37             x=y;
38             y=rem;
39             rem=x%y;
40         }
41         return y;
42     }
43 }
44
45 }
46
```

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\Algorithm\gcd.exe"

Enter The First Number

24

Enter the Second Number

12

GCD OF 24 AND 12 is 12

Process returned 0 (0x0) execution time : 11.763 s

Press any key to continue.

■

BINARY SEARCH(ITERATIVE)

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<ctype.h>
4  #define MAX 10
5  int a[MAX];
6  int flag;
7  int binary(int a[],int low,int high);
8  int num;
9  int main()
10 {
11     int n;
12     printf("Enter the Number of elements of the array\n");
13     scanf("%d",&n);
14     printf("Enter the elements of the array In Sorted order\n");
15     for(int i=0;i<n;i++)
16     {
17         scanf("%d",&a[i]);
18     }
19     printf("Enter the element You Want To Search\n");
20     scanf("%d",&num);
21     int low=0;
22     int high=n-1;
23     int ans=binary(a, low, high);
24     if(ans==0)
25     {
26         printf("Element Not Found\n");
27     }
28 }
29
30 int binary(int a[],int low,int high)
31 {
32
33     int mid=(low+high)/(2);
34     if(a[mid]==num)
35     {
36         flag=1;
37         printf("First Occurance of Element Found At Positon %d",mid);
38         return 1;
39     }
40     else if(num<a[mid])
41     {
42         high=mid-1;
43         binary(a, low, high);
44     }
45     else if(num>a[mid])
46     {
47         low=mid+1;
```

```
48     binary(a, low, high);
```

```
49 }
```

```
50 else
```

```
51 {
```

```
52     return 0;
```

```
53 }
```

```
54 }
```

```
55
```

```
56
```

```
57
```

```
58
```

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\Algorithm\binarysearch.exe"

Enter the Number of elements of array

5

Enter the elements of array In Sorted Order

25

30

35

40

45

Enter the element To Be searched

40

Element Found At Position 3

Process returned 0 (0x0) execution time : 17.721 s

Press any key to continue.

BINARY SEARCH(RECURSIVE)

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<ctype.h>
4  #define MAX 10
5  int a[MAX];
6  int flag;
7  int binary(int a[],int low,int high);
8  int num;
9  int main()
10 {
11     int n;
12     printf("Enter the Number of elements of the array\n");
13     scanf("%d",&n);
14     printf("Enter the elements of the array In Sorted order\n");
15     for(int i=0;i<n;i++)
16     {
17         scanf("%d",&a[i]);
18     }
19     printf("Enter the element You Want To Search\n");
20     scanf("%d",&num);
21     int low=0;
22     int high=n-1;
23     int ans=binary(a, low, high);
24     if(ans==0)
25     {
26         printf("Element Not Found\n");
27     }
28
29 }
```



```
30 int binary(int a[],int low,int high)
31 {
32
33     int mid=(low+high)/(2);
34     if(a[mid]==num)
35     {
36         flag=1;
37         printf("First Occurance of Element Found At Positon %d",mid);
38         return 1;
39     }
40     else if(num<a[mid])
41     {
42         high=mid-1;
43         binary(a, low,high);
44     }
45     else if(num>a[mid])
46     {
47         low=mid+1;
48         binary(a, low,high);
49     }
50     else
51     {
52         return 0;
53     }
54 }
```

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\Algorithm\binarysearch.exe"

Enter the Number of elements of array

5

Enter the elements of array In Sorted Order

25

30

35

40

45

Enter the element To Be searched

40

Element Found At Position 3

Process returned 0 (0x0) execution time : 17.721 s

Press any key to continue.

LINEAR SEARCH(ITERATIVE)

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<ctype.h>
4  int main()
5  {
6      int num;
7      int flag=1;
8      int n;
9      printf("Enter the Number of elements of array\n");
10     scanf("%d",&n);
11     int a[n];
12     printf("Enter the elements of array\n");
13     for(int i=0;i<n;i++)
14     {
15         scanf("%d",&a[i]);
16     }
17     printf("Enter the element To Be searched\n");
18     scanf("%d",&num);
19     for(int i=0;i<n;i++)
20     {
21         if(a[i]==num)
22         {
23             flag=0;
24             printf("First Occurance of the element is found at position %d" + i);
25             break;
26         }
27     }
28     if(flag==1)
29     {
30         printf("Element Not Found\n");
31     }
32
33 }
```

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\Algorithm\linearsrch.exe"

Enter the Number of elements of array

5

Enter the elements of array

25

30

35

40

45

Enter the element To Be searhced

35

First Occurance of the element is found at position 2

Process returned 0 (0x0) execution time : 19.607 s

Press any key to continue.

LINEAR SEARCH(RECURSIVE)

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<ctype.h>
4  #define MAX 10
5  int a[MAX];
6  int n;
7  int i;
8  int num;
9  void search(int a[]);
10 int main()
11 {
12     printf("Enter the Number Of Elements of the array\n");
13     scanf("%d",&n);
14     printf("Enter the elements of the array\n");
15     for(int i=0;i<n;i++)
16     {
17         scanf("%d",&a[i]);
18     }
19     printf("Enter the serach element\n");
20     scanf("%d",&num);
21     search(a);
22 }
23 void search(int a[])
24 {
25     int flag=1;
26     if(a[i]==num)
27     {
28         flag=0;
29         printf("Element Found At Position %d",(i+1));
30         return;
31     }
32     else
33     {
34         i++;
35         search(a);
36     }
37     if(flag==1)
38     {
39         printf("Element Not Found\n");
40         return;
41     }
42 }
43 }
```

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\Algorithm\linearsrch.exe"

Enter the Number of elements of array

5

Enter the elements of array

25

30

35

40

45

Enter the element To Be searhced

35

First Occurance of the element is found at position 2

Process returned 0 (0x0) execution time : 19.607 s

Press any key to continue.

SELECTION SORT

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

```
void swap(int *xp, int *yp)
```

```
{  
    int temp = *xp;  
    *xp = *yp;  
    *yp = temp;  
}
```

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

```
{  
    int i, j, min_idx;  
  
    for (i = 0; i < n-1; i++)  
    {  
        min_idx = i;  
        for (j = i+1; j < n; j++)  
            if (arr[j] < arr[min_idx])  
                min_idx = j;  
    }
```

```
min_idx = j;
```

```
swap(&arr[min_idx], &arr[i]);
```

```
}
```

```
void printArray(int arr[], int size)
```

```
{
```

```
    int i;
```

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

```
        printf("%d ", arr[i]);
```

```
    printf("\n");
```

```
}
```

```
int main()
```

```
{
```

```
    int n;
```

```
    printf("Enter the Number Of The Elements of array\n");
```

```
    scanf("%d", &n);
```

```
    int arr[n];
```



```
for (i=0; i < size; i++)  
    printf("%d ", arr[i]);  
printf("\n");  
}
```

```
int main()  
{  
    int n;  
    printf("Enter the Number Of The Elements of array\n");  
    scanf("%d",&n);  
    int arr[n];  
    printf("Enter the elements of the array\n");  
    for(int i=0;i<n;i++)  
        scanf("%d",&arr[i]);  
    selectionSort(arr, n);  
    printf("Sorted array: \n");  
    printArray(arr, n);  
    return 0;  
}
```

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\Algorithm\selectiosort.exe"

Enter the Number Of The Elements of array

5

Enter the elements of the array

1

4

0

6

8

Sorted array:

0 1 4 6 8

Process returned 0 (0x0) execution time : 12.485 s

Press any key to continue.

■

BUBBLE SORT

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

```
void swap(int *xp, int *yp)
```

```
{  
    int temp = *xp;  
    *xp = *yp;  
    *yp = temp;  
}
```

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

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

```
void printArray(int arr[], int size)
```

```
{  
    int i;  
    for (i=0; i < size; i++)  
        printf("%d ", arr[i]);  
    printf("\n");  
}
```

```
int main()
```

```
{  
    int n;  
    printf("Enter the Number Of The Elements of array\n");  
    scanf("%d",&n);  
    int arr[n];  
    printf("Enter the elements of the array\n");  
    for(int i=0;i<n;i++)  
        scanf("%d",&arr[i]);  
    bubbleSort(arr, n);  
    printf("Sorted array: \n");  
    printArray(arr, n);  
    return 0;  
}
```

"C:\Users\Shreehari Kulkarni\OneDrive\Desktop\Algorithm\selectiosort.exe"

Enter the Number Of The Elements of array

5

Enter the elements of the array

1

4

0

6

8

Sorted array:

0 1 4 6 8

Process returned 0 (0x0) execution time : 12.485 s

Press any key to continue.

■

DEPTH FIRST SEARCH

```
1  #include<stdio.h>
2  #include<time.h>
3  void DFS(int);
4  int G[10][10],visited[10],n;
5  void main()
6  {
7      int i,j;
8      clock_t start,end;
9      printf("Enter number of vertices:");
10
11      scanf("%d",&n);
12
13
14      printf("\nEnter adjacency matrix of the graph:");
15
16      for(i=0;i<n;i++)
17      for(j=0;j<n;j++)
18          scanf("%d",&G[i][j]);
19
20
21      for(i=0;i<n;i++)
22          visited[i]=0;
23      start=clock();
24      DFS(0);
25      end=clock();
26      printf("\nTime Taken Is %f\n",((double)((end-start)/(CLOCKS_PER_SEC))));
27  }
28
29 void DFS(int i)
30 {
31     int j;
32     printf("\n%d",i);
33     visited[i]=1;
34
35     for(j=0;j<n;j++)
36         if(!visited[j]&&G[i][j]==1)
37             DFS(j);
38 }
```

Enter number of vertices:5

Enter adjacency matrix of the graph:

0 1 1 1 1

1 0 1 1 1

1 1 0 1 1

1 1 1 0 1

1 1 1 1 0

0

1

2

3

4

Time Taken Is 0.000000

Process returned 24 (0x18) execution time : 37.984 s

Press any key to continue.

■

TOWER OF HANOI

```
1  #include<stdio.h>
2  #include<math.h>
3  #include<string.h>
4  #include<time.h>
5  void toh(char src,char dest,char sp,int n);
6  int main()
7  {
8      clock_t start,end;
9      int a[]={3};
10     double b[10];
11     for(int i=0;i<1;i++)
12     {
13         start=clock();
14         printf("\n *****For N=%d*****\n",a[i]);
15         toh('A','C','B',a[i]);
16         end=clock();
17         b[i]=((float)((end-start)/(CLOCKS_PER_SEC)));
18     }
19     for(int i=0;i<1;i++)
20     {
21         printf("\nTIME TAKEN FOR %d = %f\n",a[i],b[i]);
22         printf("No Of Steps Taken is %f\n",(pow(2,a[i])-1));
23     }
24 }
25
26 void toh(char src,char dest,char sp,int n)
27 {
28
29     if(n==1)
30     {
31         printf("Move From %c to %c\n",src,dest);
32         return;
33     }
34     else
35     {
36         toh(src,sp,dest,n-1);
37         toh(src,dest,sp,1);
38         toh(sp,dest,src,n-1);
39     }
40 }
```


Enter the number of disks: 4

Move disk 1 from S to T

Move disk 2 from S to D

Move disk 1 from T to D

Move disk 3 from S to T

Move disk 1 from D to S

Move disk 2 from D to T

Move disk 1 from S to T

Move disk 4 from S to D

Move disk 1 from T to D

Move disk 2 from T to S

Move disk 1 from D to S

Move disk 3 from T to D

Move disk 1 from S to T

Move disk 2 from S to D

Move disk 1 from T to D

Process returned 23 (0x17) execution time : 2.773 s

Press any key to continue.

INSERTION SORT

```
1  #include<stdio.h>
2  #include<time.h>
3  void sort(int a[],int n);
4  int main()
5  {
6      clock_t t;
7      int n;
8      printf("\nEnter the Number Of Elements Of The Array\n");
9      scanf("%d",&n);
10     int a[n];
11     printf("Enter the elements of the array\n");
12     for(int i=0;i<n;i++)
13         scanf("%d",&a[i]);
14     t=clock();
15     sort(a,n);
16     t=clock()-t;
17     double time_taken=((double)t)/CLOCKS_PER_SEC;
18     printf("Time Taken =%f\n",time_taken);
19
20     printf("Final Sorted Order Is\n");
21     for(int i=0;i<n;i++)
22     {
23         printf("%d\t",a[i]);
24     }
25
26 }
27 void sort(int a[] ,int n)
28 {
29     int v,j;
30     for(int i=1;i<=n-1;i++)
31     {
32         v=a[i];
33         j=i-1;
34         while(j>=0 && a[j]>v)
35         {
36             a[j+1]=a[j];
37             j=j-1;
38         }
39         a[j+1]=v;
40
41     }
42 }
```

Enter the Number Of Elements Of The Array

5

Enter the elements of the array

10

9

8

7

6

Time Taken =0.000000

Final Sorted Order Is

6 7 8 9 10

Process returned 0 (0x0) execution time : 5.897 s

Press any key to continue.

BFS

```
1  #include<stdio.h>
2  #include<math.h>
3  #include<stdlib.h>
4  #include<string.h>
5  int q[100];
6  int visited[100];
7  int adj[20][20];
8  int n;
9  void enqueue(int v);
10 int dequeue();
11 int front=-1;
12 int rear=-1;
13 void enqueue(int v)
14 {
15     if(front==-1 && rear==-1)
16     {
17         front=rear=0;
18     }
19     if(rear==n-1)
20     {
21         printf("Queue Full\n");
22         return;
23     }
24     q[rear]=v;
25     rear++;
26 }
27 int dequeue()
28 {
29     int val;
30     if(front==-1 || front>rear)
31     {
32         //printf("Queue Underflow\n");
33         return -1;
34     }
35     val=q[front];
36     if(front==rear || front>rear)
37     {
38         front=-1;
39         rear=-1;
40     }
41     front++;
42     return val;
43 }
44 void bfs(int v)
```

```

45 {
46     for(int i=0;i<n;i++)
47     {
48         if(adj[v][i]==1 && visited[i]==0)
49         {
50             enqueue(i);
51             printf("%d\t",i);
52             visited[i]=1;
53         }
54     }
55     int val=dequeue();
56
57     if(val!=-1)
58     {
59         bfs(val);
60     }
61     else
62     {
63         return;
64     }
65 }
66 int main()
67 {
68     int flag=1;
69     int v;
70     printf("Enter the Number of the vertex\n");
71     scanf("%d",&n);
72     printf("Enter the Entries Of The Adjacent Matrix\n");
73     for(int i=0;i<n;i++)
74     {
75         for(int j=0;j<n;j++)
76         {
77             scanf("%d",&adj[i][j]);
78         }
79     }
80     printf("Enter the Starting Vertex\n");
81     scanf("%d",&v);
82     printf("BREADTH ORDER TRAVERSAL FOR FOREST 1 IS\n");
83     printf("%d\t",v);
84     visited[v]=1;
85     bfs(v);
86
87     printf("\nBREADTH ORDER TRAVERSAL FOR FOREST 2 IF IT EXISTS\n");
88     for(int i=0;i<n;i++)

```

```
87     printf( "BREADTH FIRST TRAVEL FOR FOREST & IF IT EXISTS\n" );
88     for(int i=0;i<n;i++)
89     {
90         if(visited[i]==0)
91         {
92             flag=0;
93             printf("%d\t",i);
94             visited[i]=1;
95             bfs(i);
96             break;
97         }
98     }
99     if(flag==1)
100     {
101         printf("\nGRAPH IS CONNECTED\n");
102     }
103
104
105 }
```

Enter the Number of the vertex

10

Enter the Entries Of The Adjacent Matrix

0 0 1 1 1 0 0 0 0 0

0 0 0 0 1 1 0 0 0 0

1 0 0 1 0 1 0 0 0 0

1 0 1 0 0 0 0 0 0 0

1 1 0 0 0 1 0 0 0 0

0 1 1 0 1 0 0 0 0 0

0 0 0 0 0 0 0 1 0 1

0 0 0 0 0 0 1 0 1 0

0 0 0 0 0 0 0 1 0 1

0 0 0 0 0 0 1 0 1 0

Enter the Starting Vertex

0

BREADTH ORDER TRAVERSAL FOR FOREST 1 IS

0 2 3 4 5 1

SBREADTH ORDER TRAVERSAL FOR FOREST 2 IF IT EXISTS

6 7 9 8

Process returned 0 (0x0) execution time : 103.788 s

Press any key to continue.

TOPOLOGICAL ORDER

```
1  #include<stdio.h>
2  #include<math.h>
3  #include<string.h>
4  int front=-1;
5  int rear=-1;
6  void push(int);
7  int pop();
8  int st[10];
9  int adj[10][10];
10 int indegree[10];
11 int t[10];
12 int k;
13 int n;
14
15 void push(int x)
16 {
17     if(front==-1 && rear==-1)
18         front=rear=0;
19
20     else if(rear==n-1)
21         return;
22     else
23     {
24         rear++;
25     }
26     st[rear]=x;
27 }
28 int pop()
29 {
30     int val;
31     if(front==-1 || front>rear)
32     {
33         return -1;
34     }
35     val=st[front];
36     if(front==rear || front>rear)
37     {
38         front=-1;
39         rear=-1;
40     }
41     else
42     {
43         front++;
44     }
45 }
```



```
46     return val;
47 }
48 }
49 int main()
50 {
51
52     int sum=0;
53     printf("Enter The Number Of Vertices \n");
54     scanf("%d",&n);
55     printf("Enter The Adjacency Matrix\n");
56     for(int i=0;i<n;i++)
57     {
58         for(int j=0;j<n;j++)
59         {
60             scanf("%d",&adj[i][j]);
61         }
62     }
63     for(int i=0;i<n;i++)
64     {
65         sum=0;
66         for(int j=0;j<n;j++)
67         {
68             sum=sum + adj[j][i];
69         }
70         indegree[i]=sum;
71     }
72     for(int i=0;i<n;i++)
73     {
74         if(indegree[i]==0)
75         {
76             push(i);
77         }
78     }
79     while(front!=-1)
80     {
81         int u=pop();
82         if(u==-1)
83             break;
84         t[k]=u;
85         k++;
86         for(int j=0;j<n;j++)
87         {
88             if(adj[u][j]==1)
89             {
```

```
89         {
90             ((indegree[j])--);
91             if(indegree[j]==0)
92             {
93                 push(j);
94             }
95         }
96     }
97 }
98 printf("Final Solution Is \n");
99 for(int i=0;i<k;i++)
100 {
101     printf("%d\t",t[i]);
102 }
103 }
```

Enter The Number Of Vertices

5

Enter The Adjacency Matrix

0 0 1 0 0

0 0 1 0 0

0 0 0 1 1

0 0 0 0 1

0 0 0 0 0

Final Solution Is

0 1 2 3 4

Process returned 0 (0x0) execution time : 25.321 s

Press any key to continue.

■

MERGE SORT

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<time.h>
4  void mergesort(int left[],int right[],int a[],int nl,int nr);
5  void split(int n,int a[])
6  {
7
8      int mid,i,j;
9      if(n<2)
10         return;
11     mid=(n)/2;
12     int left[mid];
13     int right[n-mid];
14     for(i=0;i<mid;i++)
15     {
16         left[i]=a[i];
17     }
18     for(j=mid;j<n;j++)
19     {
20         right[j-mid]=a[j];
21     }
22     split((sizeof(left))/(sizeof(int)), left);
23     split((sizeof(right))/(sizeof(int)), right);
24     mergesort(left,right,a, (int)((sizeof(left))/sizeof(int)), (int)((sizeof(right))/sizeof(int)));
25 }
26 void mergesort(int left[],int right[],int a[],int nl,int nr)
27 {
28     int i=0;
29     int j=0;
30     int k=0;
31     while(i<nl&& j<nr)
32     {
33         if(left[i]<right[j])
34         {
35             a[k]=left[i];
36             i++;
37         }
38         else
39         {
40             a[k]=right[j];
41             j++;
42         }
43         k++;
44     }
45     while(i<nl)
```

```
47     a[k]=left[i];
48     i++;
49     k++;
50 }
51 while(j<nr)
52 {
53     a[k]=right[j];
54     j++;
55     k++;
56 }
57 }
58 int main()
59 {
60     clock_t t;
61     int n;
62     printf("Enter The Size Of The Array\n");
63     scanf("%d",&n);
64     int a[n];
65     printf("Enter The Elements of the array\n");
66     for(int i=0;i<n;i++)
67     {
68         //scanf("%d",&a[i]);
69         a[i]=rand();
70     }
71     t=clock();
72     split(n,a);
73     t=clock()-t;
74     double time_taken=((double)t)/CLOCKS_PER_SEC;
75     printf("Time Taken =%f\n",time_taken);
76     printf("Final Sorted Order Is\n");
77     for(int i=0;i<n;i++)
78     {
79         printf("%d\t",a[i]);
80     }
81
82 }
```

Enter The Elements of the array

23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

Time Taken =0.000006

Final Sorted Order Is

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
8	19	20	21	22	23											

...Program finished with exit code 0

Press ENTER to exit console.

QUICK SORT

```
1  #include<stdio.h>
2  #include<math.h>
3  #include<string.h>
4  #include<stdlib.h>
5  #include<time.h>
6  void swap(int *a,int *b)
7  {
8      int temp;
9      temp=*a;
10     *a=*b;
11     *b=temp;
12 }
13 int partition(int a[],int start,int end)
14 {
15     int pivot=a[end];
16     int pindex=start;
17     for(int i=start;i<end;i++)
18     {
19         if(a[i]<=pivot)
20         {
21             swap(&a[pindex],&a[i]);
22             pindex++;
23         }
24     }
25     swap(&a[pindex],&a[end]);
26     return pindex;
27 }
28 void quicksort(int a[],int start,int end)
29 {
30     //printf("HELoo\n");
31     if(start<end)
32     {
33         int pi=partition(a,start,end);
34         quicksort(a,start,pi-1);
35         quicksort(a,pi+1,end);
36     }
37 }
38 int main()
39 {
40     int n,option;
41     printf("Enter the number of the elements the array\n");
42     scanf("%d",&n);
43     int a[n];
44     printf("Enter The Elements of the array\n");
```

```

38 int main()
39 {
40     int n,option;
41     printf("Enter the number of the elements the array\n");
42     scanf("%d",&n);
43     int a[n];
44     printf("Enter The Elements of the array\n");
45     printf("1:Input From User\n");
46     printf("2: Using Random Fucntion\n");
47     printf("Enter Your Option\n");
48     scanf("%d",&option);
49     printf("\n");
50     switch(option)
51     {
52     case 1:
53         printf("Enter The Elements\n");
54         for(int i=0;i<n;i++)
55         {
56             scanf("%d",&a[i]);
57         }
58         break;
59     case 2:
60         printf("Elements Taken In Random Order\n");
61         for(int i=0;i<n;i++)
62         {
63             a[i]=rand();
64         }
65         break;
66     }
67     clock_t t;
68     t=clock();
69     quicksort(a,0,n-1);
70     t=clock() - t;
71     printf("Elements of the array are\n");
72     double time_taken=((double)t)/CLOCKS_PER_SEC;
73     for(int i=0;i<n;i++)
74     {
75         printf("%d\t",a[i]);
76     }
77     printf("\n Time taken for %d = %f seconds \n",n,time_taken);
78 }

```


Enter the number of the elements the array

5

Enter The Elements of the array

1:Input From User

2: Using Random Fucntion

Enter Your Option

1

Enter The Elements

9

8

4

3

2

Elements of the array are

2 3 4 8 9

Time taken for 5 = 0.000000 seconds

Process returned 0 (0x0) execution time : 10.390 s

Press any key to continue.

HEAP SORT

```
1  #include<stdio.h>
2  #include<math.h>
3  void swap(int *x,int *y)
4  {
5      int temp=*x;
6      *x=*y;
7      *y=temp;
8  }
9  void heapify(int a[],int n)
10 {
11     for(int i=n/2;i>=1;i--)
12     {
13         int k=i;
14         int v=a[k];
15         int heap=0;
16         while(!heap && 2*k<=n)
17         {
18             int j=2*k;
19             if(j<n)
20             {
21                 if(a[j]<a[j+1])
22                 {
23                     j=j+1;
24                 }
25             }
26             if(v>a[j])
27                 heap=1;
28             else
29             {
30                 a[k]=a[j];
31                 k=j;
32             }
33             a[k]=v;
34         }
35     }
36 }
```

```
36     },
37     int main()
38     {
39         int n;
40         printf("Enter the number of elements of the array\n");
41         scanf("%d",&n);
42         int a[n+1];
43         printf("Enter the elements of the array\n");
44         for(int i=1;i<(n+1);i++)
45         {
46             scanf("%d",&a[i]);
47         }
48         heapify(a,n);
49
50         printf("\n");
51         for(int i=n;i>=1;i--)
52         {
53             swap(&a[1],&a[i]);
54             heapify(a,i-1);
55         }
56         printf("AFTER SORTING THE ARRAY IS\n");
57         for(int i=1;i<n+1;i++)
58             printf("%d\t",a[i]);
59
60
61     }
```

KnapSack

```
1  #include<stdio.h>
2  #include<math.h>
3  int max(int a,int b)
4  {
5      if(a>b)
6          return a;
7      else
8          return b;
9  }
10
11 int main()
12 {
13     int cap,n;
14     printf("Enter the knapsack capacity\n");
15     scanf("%d",&cap);
16     printf("Enter the Number of instances\n");
17     scanf("%d",&n);
18     int w[n+1];
19     int v[n+1];
20     printf("Enter the weight of the Instances\n");
21     for(int i=1;i<=n;i++)
22     {
23         printf("Enter the weight of the instance %d\n",i);
24         scanf("%d",&w[i]);
25     }
26     printf("Enter the Value of the instances\n");
27     for(int i=1;i<=n;i++)
28     {
29         printf("Enter the value of the instance %d\n",i);
30         scanf("%d",&v[i]);
31     }
32     int a[n+1][cap+1];
33     for(int i=0;i<=n;i++)
34     {
35         for(int j=0;j<=cap;j++)
36         {
37             a[i][j]=0;
38         }
39         printf("\n");
40     }
41     int optimal=0;
42     int ins=0;
43     for(int i=1;i<=n;i++)
44     {
45         for(int j=1;j<=cap;j++)
46         {
47             if(j-w[i]>=0)
```

```

49         a[i][j]=max(a[i-1][j],v[i]+a[i-1][j-w[i]]);
50         if(a[i][j]>optimal)
51         {
52             optimal=a[i][j];
53             ins=i;
54         }
55     }
56     else
57     {
58         a[i][j]=a[i-1][j];
59     }
60 }
61 }
62 printf("FINAL SOLUTION MATRIX IS\n");
63 for(int i=0;i<n+1;i++)
64 {
65     for(int j=0;j<=cap;j++)
66     {
67         printf("%d\t",a[i][j]);
68     }
69     printf("\n");
70 }
71 printf("Optimal Solution Is %d\n",optimal);
72 printf("Optimal solution includes following instances\n");
73 printf("%d\t",ins);
74 v[ins]=0;
75 int wl=cap-w[ins];
76 int previns=ins;
77 while(wl>=0)
78 {
79     optimal=0;
80     for(int i=1;i<=n;i++)
81     {
82
83         if(v[i]!=0)
84         {
85             if(a[i][wl]>optimal)
86             {
87                 optimal=a[i][wl];
88                 ins=i;
89             }
90         }
91     }
92 }
93 printf("%d\t",ins);
94 v[ins]=0;
95 wl=wl-w[ins];

```

Enter the no. of items: 5

Enter weight of the each item:

3 4 1 5 2

Enter profit of each item:

1 5 2 6 3

Enter the knapsack's capacity: 7

the output is:

0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	0	1	5	5	5	6
0	2	2	2	5	7	7	7
0	2	2	2	5	7	8	8
0	2	3	5	5	7	8	10

the optimal solution is 10

the solution vector is:

0 1 1 0 1

Process returned 13 (0xD) execution time : 65.211 s

Press any key to continue.

FLOYD

```
2  #include<math.h>
3
4  int min(int a,int b)
5  {
6      if(a<b)
7      {
8          return a;
9      }
10     else
11     {
12         return b;
13     }
14 }
15 int main()
16 {
17     int n;
18     printf("Enter the number of vertices\n");
19     scanf("%d",&n);
20     int a[n+1][n+1];
21     printf("Enter the Distance matrix\n");
22     printf("GIVE THE DISTANC AS 9999 If there is no edge\n");
23     for(int i=1;i<=n;i++)
24     {
25         for(int j=1;j<=n;j++)
26         {
27             scanf("%d",&a[i][j]);
28         }
29     }
30     int i,j,k;
31     for(k=1;k<=n;k++)
32     {
33         for(i=1;i<=n;i++)
34         {
35             for(int j=1;j<=n;j++)
36             {
37                 a[i][j]=min(a[i][j],(a[i][k]+a[k][j]));
38             }
39         }
40     }
41     printf("ALL PAIR SHORTEST PATH DISTANCE MATRIX IS\n");
42     for(int i=1;i<=n;i++)
43     {
44         for(int j=1;j<=n;j++)
45         {
46             printf("%d\t",a[i][j]);
47         }
48         printf("\n");
49     }
50 }
```

Enter number of vertices

4

Enter the matrix:

0 5 99999 10

99999 0 3 99999

99999 99999 0 1

99999 0 99999 0

Output

0	5	8	9
INF	0	3	4
INF	1	0	1
INF	0	3	0

Process returned 0 (0x0) execution time : 55.704 s

Press any key to continue.

WARSHALL

```
1  #include<stdio.h>
2  #include<math.h>
3
4
5  int main()
6  {
7      int n;
8      printf("Enter the number of vertices\n");
9      scanf("%d",&n);
10     int a[n+1][n+1];
11     printf("Enter the adjacency matrix\n");
12     for(int i=1;i<=n;i++)
13     {
14         for(int j=1;j<=n;j++)
15         {
16             scanf("%d",&a[i][j]);
17         }
18     }
19     int i,j,k;
20     for(k=1;k<=n;k++)
21     {
22         for(i=1;i<=n;i++)
23         {
24             for(int j=1;j<=n;j++)
25             {
26                 if(a[i][j]==1)
27                 {
28                     a[i][j]=1;
29                 }
30                 if(a[i][k]==1 && a[k][j]==1)
31                 {
32                     a[i][j]=1;
33                 }
34             }
35         }
36     }
37     printf("TRANSITIVE CLOSURE IS\n");
38     for(int i=1;i<=n;i++)
39     {
40         for(int j=1;j<=n;j++)
41         {
42             printf("%d\t",a[i][j]);
43         }
44         printf("\n");
45     }
46 }
```

Enter number of vertices

5

Enter the matrix:

0 0 1 0 0

0 0 1 0 0

0 0 0 1 1

0 0 0 0 0

0 0 0 1 0

Output:

closure of the given graph

1 0 1 1 1

0 1 1 1 1

0 0 1 1 1

0 0 0 1 0

0 0 0 1 1

Process returned 0 (0x0) execution time : 74.687 s

Press any key to continue.

DIJKSTRA

```
2  #include<limits.h>
3  #include<stdbool.h>
4  #define V 5
5  int minKey(int key[],bool mstset[])
6  {
7      int min,minIndex;
8      min=INT_MAX;
9      for(int i=0;i<V;i++)
10     {
11         if(mstset[i]==false && key[i]<min)
12         {
13             min=key[i];
14             minIndex=i;
15         }
16     }
17     return minIndex;
18 }
19 void printmst(int key[])
20 {
21     int sum=0;
22     printf("Distance From Source\n");
23     for(int i=0;i<V;i++)
24     {
25         printf("%d-%d\t%d\n",0,i,key[i]);
26     }
27 }
28 }
29 void primst(int graph[V][V])
30 {
31     int parent[V];
32     int key[V];
33     bool mstset[V];
34     int sum=0;
35     for(int i=0;i<V;i++)
36     {
37         parent[i]=0;
38         key[i]=INT_MAX;
39         mstset[i]=false;
40     }
41     key[0]=0;
42     parent[0]=-1;
43     for(int count=0;count<V-1;count++)
44     {
45         int u=minKey(key,mstset);
```

```

46     mstset[u]=true;
47     for(int v=0;v<V;v++)
48     {
49         if(graph[u][v] && mstset[v]==false && key[u]!=INT_MAX && key[u]+graph[u][v]<key[v])
50         {
51             parent[v]=u;
52             key[v]=key[u] + graph[u][v];
53         }
54     }
55 }
56 printmst(key);
57 }
58 int main()
59 {
60     int graph[V][V];
61     printf("Enter the Distance Matrix\n");
62     for(int i=0;i<V;i++)
63     {
64         for(int j=0;j<V;j++)
65         {
66             scanf("%d",&graph[i][j]);
67         }
68     }
69     primst(graph);
70 }
71
72

```

Enter the Distance Matrix

```
0 3 0 7 0
3 0 4 2 0
0 4 0 5 6
7 2 5 0 4
0 0 6 4 0
```

Distance From Source

```
0-0    0
0-1    3
0-2    7
0-3    5
0-4    9
```

Process returned 0 (0x0) execution time : 32.079 s

Press any key to continue.

PRIMS

```
2  #include<limits.h>
3  #include<stdbool.h>
4  #define V 6
5  int minKey(int key[],bool mstset[])
6  {
7      int min,minIndex;
8      min=INT_MAX;
9      for(int i=0;i<V;i++)
10     {
11         if(mstset[i]==false && key[i]<min)
12         {
13             min=key[i];
14             minIndex=i;
15         }
16     }
17     return minIndex;
18 }
19 void printmst(int parent[],int graph[V][V])
20 {
21     int sum=0;
22     printf("EDGE\tWEIGHT\n");
23     for(int i=1;i<V;i++)
24     {
25         printf("%d-%d\t%d\n",parent[i],i,graph[i][parent[i]]);
26         sum=sum+graph[i][parent[i]];
27     }
28     printf("Total Minimal Cost Is%d\n",sum);
29 }
30 }
31 void primst(int graph[V][V])
32 {
33     int parent[V];
34     int key[V];
35     bool mstset[V];
36     for(int i=0;i<V;i++)
37     {
38         //parent[i]=0;
39         key[i]=INT_MAX;
40         mstset[i]=false;
41     }
42     key[0]=0;
43     parent[0]=-1;
44     for(int count=0;count<V-1;count++)
45     {
46         int u=minKey(key,mstset);
```

```

2  #include<limits.h>
3  #include<stdbool.h>
4  #define V 6
5  int minKey(int key[],bool mstset[])
6  {
7      int min,minIndex;
8      min=INT_MAX;
9      for(int i=0;i<V;i++)
10     {
11         if(mstset[i]==false && key[i]<min)
12         {
13             min=key[i];
14             minIndex=i;
15         }
16     }
17     return minIndex;
18 }
19 void printmst(int parent[],int graph[V][V])
20 {
21     int sum=0;
22     printf("EDGE\tWEIGHT\n");
23     for(int i=1;i<V;i++)
24     {
25         printf("%d-%d\t%d\n",parent[i],i,graph[i][parent[i]]);
26         sum=sum+graph[i][parent[i]];
27     }
28     printf("Total Minimal Cost Is%d\n",sum);
29 }
30 }
31 void primst(int graph[V][V])
32 {
33     int parent[V];
34     int key[V];
35     bool mstset[V];
36     for(int i=0;i<V;i++)
37     {
38         //parent[i]=0;
39         key[i]=INT_MAX;
40         mstset[i]=false;
41     }
42     key[0]=0;
43     parent[0]=-1;
44     for(int count=0;count<V-1;count++)
45     {
46         int u=minKey(key,mstset);

```

Enter the Distance Matrix

0 3 0 0 6 5

3 0 1 0 0 4

0 1 0 6 0 4

0 0 6 0 8 5

0 0 0 8 0 2

5 4 4 5 2 0

EDGE WEIGHT

0-1 3

1-2 1

5-3 5

5-4 2

1-5 4

Total Minimal Cost Is15

Process returned 0 (0x0) execution time : 215.634 s

Press any key to continue.

■

KRUSKALL

```
2  #include<math.h>
3  int n;
4  int c[10][10];
5  void kruskal()
6  {
7      int i,j,a,b,u,v,min;
8      int mincost=0;
9      int ne=0;
10     int parent[n+1];
11     for(int i=1;i<=n;i++)
12     {
13         parent[i]=0;
14     }
15     while(ne!=n-1)
16     {
17         min=9999;
18         for(int i=1;i<=n;i++)
19         {
20             for(int j=1;j<=n;j++)
21             {
22                 if(c[i][j]<min)
23                 {
24                     min=c[i][j];
25                     u=a=i;
26                     v=b=j;
27                 }
28             }
29         }
30         while(parent[u]!=0)
31         {
32             u=parent[u];
33         }
34         while(parent[v]!=0)
35         {
36             v=parent[v];
37         }
38         if(v!=u)
39         {
40             printf("%d-%d\t%d\n",u,v,min);
41             parent[v]=u;
42             ne++;
43             mincost=mincost+min;
44         }
45         c[a][b]=c[b][a]=9999;
```

```
47     printf("\n MIN COST=%d\n",mincost);
48
49 }
50 int main()
51 {
52     printf("Enter the Number of vertices\n");
53     scanf("%d",&n);
54     printf("Enter the Cost Matrix\n");
55     for(int i=1;i<=n;i++)
56     {
57         for(int j=1;j<=n;j++)
58         {
59             scanf("%d",&c[i][j]);
60         }
61     }
62     printf("EDGE\tWeight\n");
63     kruskal();
64 }
```

Enter the Number of vertices

6

Enter the Cost Matrix

9999 3 9999 9999 6 5

3 9999 1 9999 9999 4

9999 1 9999 6 9999 4

9999 6 6 9999 8 5

6 9999 9999 8 9999 2

5 4 4 5 2 9999

EDGE Weight

2-3 1

5-6 2

1-2 3

1-5 4

4-1 5

MIN COST=15

Process returned 0 (0x0) execution time : 85.583 s

Press any key to continue.

NQUEENS

```
2 //The code snippet is
3 int n;
4 int x[10];
5 int place(int k,int i)
6 {
7     for(int j=1;j<=k-1;j++)
8     {
9         if((x[j]==i) || (abs(x[j]-i) == (abs(j-k))))
10             return 0;
11     }
12     return 1;
13 }
14 void nqueens(int k,int n)
15 {
16     for(int i=1;i<=n;i++)
17     {
18         if(place(k,i))
19         {
20             x[k]=i;
21             if(k==n)
22             {
23                 printf("\nSOLUTION ARE\n");
24                 for(int i=1;i<=n;i++)
25                 {
26                     printf("%d\t",x[i]);
27                 }
28                 printf("\n");
29             }
30             else
31             {
32                 nqueens(k+1,n);
33             }
34         }
35     }
36 }
37 int main()
38 {
39     printf("Enter the Number of the queens\n");
40     scanf("%d",&n);
41     for(int k=1;k<=n;k++)
42     {
43         nqueens(k,n);
44     }
45 }
46 }
```

Enter the Number of the queens

4

SOLUTION ARE

2 4 1 3

SOLUTION ARE

3 1 4 2

Process returned 0 (0x0) execution time : 6.565 s

Press any key to continue.

■

SUM OF SUBSETS

```
1  #include<stdio.h>
2  #include<math.h>
3  int d;
4  int n;
5  int w[10];
6  int x[10];
7  void sumofsubs(int cs,int k,int r)
8  {
9      if(k<=n)
10     {
11         x[k]=1;
12         if(cs+w[k]==d)
13         {
14             for(int i=1;i<=k;i++)
15             {
16                 if(x[i]==1)
17                     printf("%d\t",w[i]);
18             }
19             printf("\n");
20         }
21         else
22         {
23             if(cs+w[k]+w[k+1]<=d)
24                 sumofsubs(cs+w[k],k+1,r-w[k]);
25         }
26         if(cs+r-w[k]>=d && cs+w[k+1]<=d)
27         {
28             x[k]=0;
29             sumofsubs(cs,k+1,r-w[k]);
30         }
31     }
32 }
33
34 }
35 int main()
36 {
37
38     printf("Enter the Target Weight \n");
39     scanf("%d",&d);
40     printf("Enter The Number of weights\n");
41     scanf("%d",&n);
42     printf("Enter the weights\n");
43     int r=0;
44     for(int i=1;i<=n;i++)
```

```
44     sumofsubs(cs,1,r);
```

```
45     {
```

```
46         scanf("%d",&w[i]);
```

```
47         r+=w[i];
```

```
48         x[i]=0;
```

```
49         //printf("\n");
```

```
50     }
```

```
51     // printf("%d\t\n",r);
```

```
52     printf("\nSOLUTION WEIGHTS ARE\n");
```

```
53     int cs=0;
```

```
54     sumofsubs(cs,1,r);
```

```
55 }
```

Enter the Target Weight

30

Enter The Number of weights

6

Enter the weights

5

10

12

13

15

18

SOLUTION WEIGHTS ARE

5	10	15
---	----	----

5	12	13
---	----	----

12	18	
----	----	--

Process returned 0 (0x0) execution time : 12.021 s

Press any key to continue.