

```
1}}}}}}}}}}}} tower
```

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

```
#include<time.h>
```

```
void towerOfHanoi (int n, char from_rod, char to_rod, char aux_rod)
{
    if (n == 1)
    {
        printf ("\n Move disk 1 from rod %c to rod %c", from_rod, to_rod);
        return;
    }
    towerOfHanoi (n-1, from_rod, aux_rod, to_rod);
    printf ("\n Move disk %d from rod %c to rod %c", n, from_rod, to_rod);
    towerOfHanoi (n-1, aux_rod, to_rod, from_rod);
}
```

```
int main ()
{
    int n;
    printf("\nEnter the number of disks\n");
    scanf("%d",&n);
    clock_t start=clock ();
    towerOfHanoi (n, 'A','C', 'B');
    clock_t end=clock();
    printf ("\nStart time is %lf\n",(double)start);
    printf ("End time is %lf\n",(double)end);
    printf ("Total time is %lf\n",(double)(end-start));
    return 0;
}
*****
2}}}}}}}} BINARY SEARCH
```

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

```
#include <time.h>
```

```
int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);
        return binarySearch(arr, mid + 1, r, x);
    }
    return -1;
}
```

```
int main()
{
    int n,x;
    printf("Enter size\n");
    scanf("%d",&n);
    int arr[n];
    printf("Enter array elements\n");
```

```

for(int i=0;i<n;i++)
{
    scanf("%d",&arr[i]);
}
printf("Enter key\n");
scanf("%d",&x);
clock_t start=clock();
int result = binarySearch(arr, 0, n - 1, x);
clock_t end=clock();
if(result == -1)
    printf("Element is not present in array\n");
else
    printf("Element is present at index %d\n", result);
printf("\nStart time is %lf\n",(double)start);
printf("\nEnd time is %lf\n",(double)end);
printf("\nTotal time is %lf\n",(double)(end-start));
return 0;
}
*****
3}}}} MERGE SORT

```

```

#include<stdlib.h>
#include<stdio.h>
#include<time.h>

```

```

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    // Create temp arrays
    int L[n1], R[n2];
    // Copy data to temp array
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    // Merge the temp arrays
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
}

```

```

// Copy the remaining elements of L[]
while (i < n1)
{
    arr[k] = L[i];
    i++;
    k++;
}
// Copy the remaining elements of R[]
while (j < n2)
{
    arr[k] = R[j];
    j++;
    k++;
}
}
}

void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        // Finding mid element
        int m = l+(r-l)/2;
        // Recursively sorting both the halves
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);
        merge(arr, l, m, r);
    }
}

void printArray(int A[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", A[i]);
}

int main()
{
    int arr[25],n;
    printf("Enter the number of elements in the array\n");
    scanf("%d",&n);
    printf("Enter the array elements\n");
    for(int i=0;i<n;i++)
        scanf("%d",&arr[i]);
    clock_t start=clock();
    mergeSort(arr, 0, n- 1);
    clock_t end=clock();
    printf("\nSorted array is\n");
    printArray(arr, n);
    printf("\nStart time is %lf\n",(double)start);
    printf("End time is %lf\n",(double)end);
    printf("Total time is %lf\n",(double)(end-start));
    return 0;
}
*****
4}}}}}}}}}}}} QUICK SORT

#include<stdio.h>

```

```
#include<time.h>
```

```
void quicksort(int a[],int low,int high);  
int partition(int a[],int low,int high);  
void swap(int*,int*);
```

```
void quicksort(int a[],int low,int high)  
{  
    if(low<high)  
    {  
        int pi = partition(a,low,high);  
        quicksort(a,low,pi-1);  
        quicksort(a,pi+1,high);  
    }  
}
```

```
void swap(int *a,int *b)  
{  
    int c=*a;  
    *a=*b;  
    *b=c;  
}
```

```
int partition(int a[],int low,int high)  
{  
  
    int pivot=a[high];  
    int i=low-1;  
    for(int j=low;j<=high-1;j++)  
    {  
        if(a[j]<=pivot)  
        {  
            i++;  
            swap(&a[i],&a[j]);  
        }  
    }  
    swap(&a[i+1],&a[high]); return (i+1);  
}
```

```
int main()  
{  
    int a[25],n;  
    printf("Enter the number of elements in the array\n");  
    scanf("%d",&n);  
    printf("Enter the elements to be sorted\n");  
    for(int i=0;i<n;i++)  
        scanf("%d",&a[i]);  
    clock_t start=clock();  
    quicksort(a,0,n-1);  
    clock_t end=clock();  
    printf("The sorted elements are\n");  
    for(int k=0;k<=4;k++)  
    {  
        printf("%d\t",a[k]);  
    }  
}
```

```

printf("\nStart time is %lf\n", (double)start);
printf("End time is %lf\n", (double)end);
printf("Total time is %lf\n", (double)(end-start));
return 0;
}
*****
5}}}}}}}} PRIMS

```

```

#include<stdio.h>
#include<time.h>
int visited[10]={0}, cost[10][10], min, mincost=0;
int i,j,ne=1, a, b, u, v;;

int main()
{
    int num;
    printf("\n\t\t\tPrim's Algorithm");
    printf("\n\nEnter the number of nodes= ");
    scanf("%d", &num);
    printf("\nEnter the adjacency matrix\n\n");
    for(i=1; i<=num; i++)
    {
        for(j=1; j<=num; j++)
        {
            scanf("%d", &cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    }
    clock_t start=clock();
    visited[1]=1;
    while(ne < num)
    {
        for(i=1,min=999;i<=num;i++)
        for(j=1;j<=num;j++)
        if(cost[i][j]< min)
        if(visited[i]!=0)
        {
            min=cost[i][j];
            a=u=i;
            b=v=j;
        }
        printf("\n Edge %d:(%d - %d) cost:%d",ne++,a,b,min);
        mincost=mincost+min;
        visited[b]=1;
        cost[a][b]=cost[b][a]=999;
    }
    printf("\n\n\n Minimun cost=%d",mincost);
    clock_t end=clock();
    printf("\nStart time is %lf\n", (double)start);
    printf("End time is %lf\n", (double)end);
    printf("Total time is %lf\n", (double)(end-start));

    return 0;
}

```

#####

```
5(2)}} Krushkal
#include<stdio.h>
#include<time.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
void main()
{
    printf("\n\tImplementation of Kruskal's algorithm\n");
    printf("\nEnter the no. of vertices:");
    scanf("%d",&n);
    printf("\nEnter the cost adjacency matrix:\n");
    for(i=1;i<=n;i++)
    {

        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    }
    printf("The edges of Minimum Cost Spanning Tree are\n"); clock_t start=clock();
    while(ne < n)
    {
        for(i=1,min=999;i<=n;i++)
        {
            for(j=1;j <= n;j++)
            {
                if(cost[i][j] < min)
                {
                    min=cost[i][j];
                    a=u=i;
                    b=v=j;
                }
            }
        }
        u=find(u);
        v=find(v);
        if(uni(u,v))
        {
            printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);
            mincost +=min;
        }
        cost[a][b]=cost[b][a]=999;
    }
    printf("\n\tMinimum cost = %d\n",mincost);
    clock_t end=clock();
    printf("Start time is %lf\n",(double)start);
    printf("End time is %lf\n",(double)end);
    printf("Total time is %lf\n",(double)(end-start));
}
int find(int i)
```

```

{
while(parent[i])
    i=parent[i];
return i;
}
int uni(int i,int j)
{
if(i!=j)
{
parent[j]=i;
return 1;
}
return 0;
}
}
*****

```

6}}}}}}}} Floyd's

```

#include<stdio.h>
#include<time.h>
void floyd(int a[10][10], int n)
{
for(int k=0;k<n;k++)
{
for(int i=0;i<n;i++)
{
for(int j=0;j<n;j++)
{
if(a[i][j]>a[i][k]+a[k][j])
{
a[i][j]=a[i][k]+a[k][j];
}
}
}
}
}
printf("All Pairs Shortest Path is :\n");
for(int i=0;i<n;i++)
{
for(int j=0;j<n;j++)
{
printf("%d ",a[i][j]);
}
printf("\n");
}
}
}
int main()
{
int cost[10][10],n;
printf("Enter the number of vertices\n");
scanf("%d",&n);
printf("Enter the cost adjacency matrix\n");
for(int i=0;i<n;i++)
{
for(int j=0;j<n;j++)
{
scanf("%d",&cost[i][j]);
}
}
}

```

```

    }
}
clock_t start=clock();
floyd(cost,n);
clock_t end=clock();
printf("Start time is %lf\n",(double)start);
printf("End time is %lf\n",(double)end);
printf("Total time is %lf\n",(double)(end-start));
return 0;
}
*****
7}}}}}} 0-1 Knapsack

```

```

#include<stdio.h>
int max(int a, int b) { return (a > b)? a : b; }
int knapSack(int W, int wt[], int val[], int n)
{
    int i, w;
    int K[n+1][W+1];
    for (i = 0; i <= n; i++)
    {
        for (w = 0; w <= W; w++)
        {
            if (i==0 || w==0)
                K[i][w] = 0;
            else if (wt[i-1] <= w)
                K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);
            else
                K[i][w] = K[i-1][w];
        }
    }
    return K[n][W];
}
int main()
{
    int i, n, val[20], wt[20], W;

    printf("Enter number of items:");
    scanf("%d", &n);

    printf("Enter value and weight of items:\n");
    for(i = 0; i < n; ++i){
        scanf("%d%d", &val[i], &wt[i]);
    }

    printf("Enter size of knapsack:");
    scanf("%d", &W);

    printf("%d", knapSack(W, wt, val, n));
    return 0;
}
*****
8}}}}}}}} Travelling Salesman Problem

```



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

```
int ary[10][10],completed[10],n,cost=0;
```

```
void takeInput()
```

```
{  
int i,j;
```

```
printf("Enter the number of villages: ");  
scanf("%d",&n);
```

```
printf("\nEnter the Cost Matrix\n");
```

```
for(i=0;i < n;i++)  
{  
printf("\nEnter Elements of Row: %d\n",i+1);
```

```
for( j=0;j < n;j++)  
scanf("%d",&ary[i][j]);
```

```
completed[i]=0;  
}
```

```
printf("\n\nThe cost list is:");
```

```
for( i=0;i < n;i++)  
{  
printf("\n");
```

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

```
void mincost(int city)  
{  
int i,ncity;
```

```
completed[city]=1;
```

```
printf("%d-->",city+1);  
ncity = least(city);
```

```
if(ncity==999)  
{  
ncity=0;  
printf("%d",ncity+1);  
cost+=ary[city][ncity];
```

```
return;  
}
```

```
mincost(ncity);  
}
```

```

int least(int c)
{
int i,nc=999;
int min=999,kmin;

for(i=0;i < n;i++)
{
if((ary[c][i]!=0)&&(completed[i]==0))
if(ary[c][i]+ary[i][c] < min)
{
min=ary[i][0]+ary[c][i];
kmin=ary[c][i];
nc=i;
}
}

if(min!=999)
cost+=kmin;

return nc;
}

int main()
{
takeInput();

printf("\n\nThe Path is:\n");
mincost(0); //passing 0 because starting vertex

printf("\n\nMinimum cost is %d\n ",cost);
}

```

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

int i,j,m,n,c[20][20];
char x[20],y[20],b[20][20];
```

```
}
```

```
void lcs()
```

```
{
```

```
  m=strlen(x);
```

```
  n=strlen(y);
```

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

```
    c[i][0]=0;
```

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

```
    c[0][i]=0;
```

```
//c, u and l denotes cross, upward and downward directions respectively
```

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

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

```
{
```

```
  if(x[i-1]==y[j-1])
```

```
{
```

```
  c[i][j]=c[i-1][j-1]+1;
```

```
  b[i][j]='c';
```

```
}
```

```
else if(c[i-1][j]>=c[i][j-1])
```

```
{
```

```
  c[i][j]=c[i-1][j];
```

```
  b[i][j]='u';
```

```
}
```

```
else
```

```
{
```

```
  c[i][j]=c[i][j-1];
```

```
  b[i][j]='l';
```

```
}
```

```
}
```

```
}
```

```
int main()
```

```
{
```

```
  printf("Enter 1st sequence:");
```

```
  scanf("%s",x);
```

```
  printf("Enter 2nd sequence:");
```

```
  scanf("%s",y);
```

```
  printf("\nThe Longest Common Subsequence is ");
```

```
  lcs();
```

```
  print(m,n);
```

```
  return 0;
```

```
}
```