**1.program to add elements in an array using dnc approach**

#include<stdio.h>

#include<stdio.h>

int add(int ,int ,int \*);

int main()

{

int \*a,n,i,sum,mid;

printf("\NEnter the no.of.elements: ");

scanf("%d",&n);

a=(int \*)malloc(n\*sizeof(int));

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

scanf("%d",a+i);

sum=add(0,n-1,a);

printf("\nSum=%d",sum);

return 0;

}

int add(int low,int high,int \*a)

{

int mid;

if(high==low)

return a[low];

mid=(low+high)/2;

return add(low,mid,a)+add(mid+1,high,a);

}

**2.MERGE SORT**

#include<stdio.h>

int arr[8]={14, 62, 22, 43, 35, 64, 57, 18},mid;

void main()

{

int i;

merge\_sort(arr, 0, 7);

printf("Sorted array:");

for(i = 0; i < 8; i++)

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

getch();

}

int merge\_sort(int arr[],int low,int high)

{

printf("\nmerge\_sort initialization\n");

if(low < high)

{

mid = (low + high) / 2;

// Divide and Conquer

merge\_sort(arr, low, mid);

printf("\n merge\_sort first\n");

merge\_sort(arr, mid + 1, high);

printf("\n merge\_sort second\n");

// Combine

merge(arr, low, mid, high);

printf("\nmerging\n");

}

return 0;

}

int merge(int arr[], int l, int m, int h)

{

int arr1[10], arr2[10];

int n1, n2, i, j, k;

n1 = m - l + 1;

n2 = h - m;

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

arr1[i] = arr[l + i];

for(j = 0; j < n2; j++)

arr2[j] = arr[m + j + 1];

arr1[i] = 9999;

arr2[j] = 9999;

i = 0;

j = 0;

for(k = l; k <= h; k++)

{

if(arr1[i] <= arr2[j])

arr[k] = arr1[i++];

else

arr[k] = arr2[j++];

}

return 0;

}

**3.QUICK SORT**

#include<stdio.h>

#include<conio.h>

void qsort(int arr[20], int fst, int last);

int main()

{

int arr[30];

int i,size;

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

scanf("%d",&size);

printf("Enter total %d elements : \n",size);

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

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

qsort(arr,0,size-1);

printf("Quick sorted elements are as : \n");

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

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

getch();

return 0;

}void qsort(int arr[20], int fst, int last)

{

int i,j,pivot,tmp;

if(fst<last)

{

pivot=fst;

i=fst;

j=last;

while(i<j)

{

while(arr[i]<=arr[pivot] && i<last)

i++;

while(arr[j]>arr[pivot])

j--;

if(i<j)

{

tmp=arr[i];

arr[i]=arr[j];

arr[j]=tmp;

}

}

tmp=arr[pivot];

arr[pivot]=arr[j];

arr[j]=tmp;

qsort(arr,fst,j-1);

qsort(arr,j+1,last);

}

}

**4.KRUSKAL ALGORITHM**

#include<stdio.h>

#include<conio.h>

#include<stdlib.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()

{

clrscr();

printf("\n\n\tImplementation of Kruskal's algorithm\n\n");

printf("\nEnter the no. of vertices\n");

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("\nThe edges of Minimum Cost Spanning Tree are\n\n");

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("\n%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);

getch();

}

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;

}

**5.PRIM’S ALGORITHM**

#include<stdio.h>

#include<conio.h>

#include<stdlib.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()

{

clrscr();

printf("\n\n\tImplementation of Kruskal's algorithm\n\n");

printf("\nEnter the no. of vertices\n");

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("\nThe edges of Minimum Cost Spanning Tree are\n\n");

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("\n%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);

getch();

}

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.DIJKSTRA’S ALGORITHM**

#include<stdio.h>

#include<conio.h>

int a,b,u,v,n,i,j,ne=1;

int visited[10]={0},min,mincost=0,cost[10][10];

void main()

{

clrscr();

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

scanf("%d",&n);

printf("\n Enter the 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;

}

visited[1]=1;

printf("\n");

while(ne<n)

{

for(i=1,min=999;i<=n;i++)

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

if(cost[i][j]<min)

if(visited[i]!=0)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

if(visited[u]==0 || visited[v]==0)

{

printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);

mincost+=min;

visited[b]=1;

}

cost[a][b]=cost[b][a]=999;

}

printf("\n Minimun cost=%d",mincost);

getch();

}

**PROGRAM -10**

**C PROGRAM TO SOLVE KNAPSACK PROBLEM USING DYNAMIC PROGRAMMING**

#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 val[] = {60, 100, 120};

int wt[] = {10, 20, 30};

int W = 50;

int n = sizeof(val)/sizeof(val[0]);

printf("\nValue = %d", knapsack(W, wt, val, n));

return 0;

}

**PROGRAM -11**

**C PROGRAM FOR MATRIX CHAIN MULTIPLICATION**

#include<stdlib.h>

#include<stdio.h>

#define MAX\_INT 10000000

int matrixMultiplication(int p[], int N);{

int L,i, j, temp;

int M[N][N];

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

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

M[i][j] = 0;

}

}

for(L=2; L<N; L++){

for(i=1; i<N-L+1; i++){

j = i+L-1;

M[i][j] = MAX\_INT;

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

temp = M[i][k] + M[k+1][j] + p[i-1] \* p[k] \* p[j];

if(temp < M[i][j]){

M[i][j] = temp;

}

}

}

}

for(i=1; i<N; i++){

for(int k=1; k<N; k++){

printf("%d ", M[i][k]);

}

printf("\n");

}

return M[1][N-1];

}

int main(){

int p [] ={10, 20, 30, 40, 30};

int n = sizeof(p)/sizeof(p[0]);

printf("%d\n", matrixMultiplication(p,n));

return 0;

}

**PROGRAM -12**

**C PROGRAM FOR LONGEST COMMON SUBSEQUENCE PROBLEM USING DYNAMIC PROGRAMMING**

#include<stdio.h>

#include<string.h>

int i,j,m,n,c[20][20];

char x[20],y[20],b[20][20];

void print(int i,int j)

{

if(i==0 || j==0)

return;

if(b[i][j]=='c')

{

print(i-1,j-1);

printf("%c",x[i-1]);

}

else if(b[i][j]=='u')

print(i-1,j);

else

print(i,j-1);

}

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;

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;

}

**OUTPUT**

Enter 1st sequence:ACFGHD

Enter 2nd sequence:ABFHD

The longest common sequence is:AFHD