Experiment-16

MST using Greedy Techniques.

→ **Aim:**

To perform Minimum Spanning tree using greedy Techniques

→ **Algorithms:**

* Give the input of number of nodes & give the matrix of nodes.
* Create for loops to get the number of nodes of matrices.
* Creating a loop in such a way that once a node is visited, then it should not be repeated, for this create another while loop for not visiting the nodes.
* Get result and End program.

→ **Program:**

```
#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];
int main(){
    Printf("\n Enter the number of nodes:");
    scanf("%d",&n);
    Printf("\n,Enter 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", nett ,a,b,min);
        minCost +=min;
        visited[b]=1;
    }
    cost[a][b]=cost[b][a]=999;
}
Printf("\n min cost = %d", minCost);
getch();
}
```

→ Output:

Enter the number of nodes: 6

→ Enter adjacency matrix:

```
0 5 6 4 0 0
5 0 1 2 0 0
6 1 0 2 5 3
4 2 2 0 0 4
0 0 5 0 0 4
0 0 3 4 4 0
```

Edge 1: (1 4) cost: 4
Edge 2: (4 2) cost: 2
Edge 3: (2 3) cost: 1
Edge 4: (3 6) cost: 3
Edge 5: (6 5) cost: 4.

Minimum Cost = 14

Experiment-17

Knapsack Problem using Greedy Tech

→ Aim:

To write a program to perform knapsack problem using greedy technique.

→ Algorithm:

* Start & declare the required inputs.
* Create void function such that the required program do taking inputs and size of knapsack bag with its capacity is being given by user.
* Create for loops for continously taking inputs upto the max value is acheived.
* Print the result and End Program.

→ Input:

```
#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=0;i<n;i++)
    x[i]=0.0;
 for(i=0;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("\n The result vector is:");
```

```
for(i=0;
  Printf("
  Printf("
}
int main(){
  float weigh
  int num, i,
  float ratio
  Printf("\n
  Scanf("%d",
  Printf("\n
  for(i=0;i<
    Scanf("
}
Printf("\n Ent
Scanf("%f"
for(i=0;i<n
{
  ratio[i]=
}
for(i=0;i<n
for(j=i+1;
  if (ratio
  temp=ro
  weight[
  temp=
  }
}
knapsack(n
return(0);
}
```

→ Program:

```
Enter no. of
Enter wts
Enter Capac
result:
maxprofit:
```

```
    for(i=0;i<n;i++)
      Printf("%.f\t",x[i]);
    Printf("\n Max profit is:%f",tp);
}
int main(){
   float weight[20],profit[20],capacity;
   int num,i,j;
   float ratio[20],temp;
   Printf("\n Enter no.of objects:");
   Scanf("%d",&num);
   Printf("\n Enter wts & profits of each object:");
   for(i=0;i<num;i++){
      Scanf("%f%f",&weight[i],&profit[i]);
}
 Printf("\n Enter capacity of knapsack:");
  Scanf("%f",&capacity);
  for(i=0;i<num;i++)
  {
    ratio[i]=profit[i]/weight[i];
  }
  for(i=0;i<num;i++){
  for(j=i+1;j<num;j++){
     if(ratio[i]<ratio[j]){
     temp=ratio[j];
     weight[j]=weight[i];
      temp=profit[j];
    }
  }
 knapsack(num,weight,Profit,capacity);
  return(0);
}
```

→ Program:

```
Enter no.of Objects:5
Enter wts & profits of each object:2 10  3 15  6 15  4 9
                                   8 20
Enter Capacity:20
result: 1.0 1.0 1.0 1.0 1.0
maxprofit: 69.00
```

Experiment-18

Optimal Binary Search tree using Dynamic Program.

→ **Aim:**

To perform the optimal binary search tree using dynamic programming.

→ **Algorithm:**

* include the limits.h file
* Create with double datatypes.
* Create a double function with optimal BST Cost.
* Create for loops with respect to optimal BST Cost.
* Create Keys in double datatypes.
* Get result & End program.

→ **Program:**

```
#include<Stdio.h>
#include<limits.h>
double SumProbabilites(double Probabilities[], int Start,
                       int end) {
    double Sum=0.0;
    for (int i= Start; i<=end; i++)
        Sum+= Probabilities[i];
    return Sum;
}
double OptimalBSTCost(double keys[], double probabilites[],
                      int n)
{
    double dp[n][n];
    for(int i=0;i<n;i++)
        dp[i][i]=Probabilites[i];
    for(int L=2; L<=n; L++)
    {
        for(int i=0; i<=n-L+1; i++)
        {
            int j=i+L-1;
            dp[i][j] =INT_MAX;
            for(int r=i; r<=j; r++)
```

```
        {
            double cost = (r>i)? dp[i][r-1]:0) + ((r<j)?
                          dp[r+1][j]:0) + SumProbabilites
                          (Probabilites, i, j);
            if (cost < dp[i][j])
                dp[i][j] = cost;
        }
    }
}
return dp[0][n-1];
}
int main()
{
    int n;
    Printf("Enter no. of keys:");
    Scanf("%d", &n);
    double keys[n], Probabilites[n];
    Printf("Enter keys & their probabilites:\n");
    for(int i=0; i<n; i++)
    {
        Scanf("%lf %lf", &keys[i], &Probabilites[i]);
    }for{
    double cost = optimalBSTcost(keys, probabilites, n);
    Prinf("Optimal BST cost: %lf \n", cost);
    return 0;
}
```

→ Output:

```
Enter no. of keys: 3
Enter keys & their probabilities:
10   15
20   24
30   10
Optimal BST cost: 74.000
```

Experiment - 19.

Binomial Coefficient using Dynamic Pgm

Aim:

To find the binomial coefficient of the given numbers using Dynamic programming

Program:

```
#include <stdio.h>
int main()
{
 int i,n=1,r=1,k=1,n1,n2,n3,binomial;
 Printf("\n Enter value of n:");
 Scanf("%d", &n1);
 Printf("\n Enter value of r:");
 Scanf("%d", &n2);
 n3 = n1-n2;
   for(i=1; i<=n1; i++)
   {
     n=n*i;
   }
   for(i=1; i<=n2; i++)
   {
     r=r*i;
   }
   for(i=1; i<=n3; i++)
   {
     k=k*i;
   }
   binomial = n/(r*k);
   if (n1<0)
   {
     Printf("\n Invalid, enter correct choice \n");
   }
   else if (n2<0)
   {
     Printf("\n Invalid; Correct choice\n");
   }
   else
   {
     Printf("\n binomial coeff is %d", binomial);
   }
 return 0;
}
```

→ Algorithm:

* Enter binomial value by creating for loops
* If entered value is less than 0, we need to Perform and print result as Invalid.
* Print result & End the program.

→ Output:

Enter value of n: 9

Enter value of r: -8

Invalid, Enter correct choice.

Experiment 20

Merge Sort.

→ Aim:

To perform merge Sorting using C programming.

→ Algorithm:

* Create duplicate copies of sub arrays to be sorted
* Maintain current index of sub arrays & main array.
* until we reach the end of either L or M, pick larger elements L & M & place them in the correct position
* when we run out of elements in either L or M, pick up remaining elements & put in a[p--r]
* Print result & stop.