Week - 09

04/07/2024

PAGE NO :
DATE :

Prims algorithm

```
#include <stdio.h>
#define INF 9999

Void Prim(int n, int cost[n][n]){
    int S[n];
    int d[n];
    int P[n];
    int i,j, min, source, sum = 0, K=0;

    min = INF;
    source =0;
    for(i=0; i<n; i++){
        for(j=0; j<n; i++){
            if(cost[i][j] !=0 && cost[i][j] <min){
                min = cost[i][j];
                source = i;
            }
        }
    }

    for(i=0; i<n; i++){
        S[i] =0;
        d[i] = cost[source][i];
        P[i] = source;
    }
    S[source] = 1;

    for(i=1; i<n; i++){
        min = INF;
        int u = -1;
        for(j=0; j<n; j++){
            if(S[j] == 0 && d[j] <min){
                min = d[j];
                u = j;
```

4/7/24

PAGE NO :
DATE :

```
        printf("(%d, %d)", u, p[u]);
        sum += cost[u][p[u]];
        s[u] = 1;

        for(j=0; j<n; j++) {
            if(s[j]==0 && cost[u][j] < d[j]) {
                d[j] = cost[u][j];
                p[j] = u;
            }
        }
    }

    if(sum >= INF) {
        printf("\n spaming tree does not Exist\n");
    } else {
        printf("\n The cost of minimum spaning tree
                is %d\n", sum);
    }

int main() {
    int n;
    printf("Enter number of vertices: ");
    scanf("%d", &n);

    int cost[n][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]);
        }
    }
    printf("\n minimum spaning tree edges:\n");
    prim(n, cost);
    return 0;
}
```

PAGE NO :
DATE :

Out Put :-

Enter the number of vertices : 4

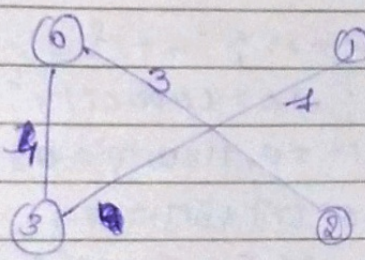
Enter the cost adjacency matrix :

0	9999	3	4
9999	0	8	7
3	8	0	6
4	7	6	0

minimum spanning tree edges:

(2, 0) (3, 0) (1, 3)

The cost of the minimum spanning tree is 14.



4+3+7 = 14

PAGE NO :
DATE :

Knapsack Algorithm

```
#include <stdio.h>

int max(int a, int b){
    return (a>b)? a :b;
}

void Knapsack(int n,int capacity, int weights[],
                int profits[]){
    int dP[n+1][capacity+1];
    int i, w;

    for(i=0; i<=n; i++){
        for(w=0; w<=capacity; w++){
            if(i==0 || w==0)
                dP[i][w]=0;
            else if(weights[i-1]<=w)
                dP[i][w] = max(profits[i-1]+
                dP[i-1][w-weights[i-1]], 0
                  dP[i-1][w]);
            else
                dP[i][w] = dP[i-1][w];
        }
    }

    int max_Profit = dP[n][capacity];
    Printf("Maximum profit: %d\n", max_Profit);
    Printf("objects selected:\n");
    w=capacity;
    for(i=n; i>0 && max_profit>0; i--){
        if(max_Profit == dP[i-1][w])
            continue;
        else{
```

```
        printf(" object %d (weight = %.d, Profit =
                %d , Profit = %.d) \n", i, weights[i-1]
                                    Profits[i-1]);
        max_profit -= profits[i-1];
        w -= weights[i-1];
    }
}
}

int main(){
    int n;
    printf("Enter number of objects: ");
    scanf("%.d", &n);

    int weights[n];
    int profits[n];

    Printf("Enter weights of the objects:\n");
    for(int i=0; i<n; i++){
        scanf("%d", &weights[i]);
    }
    printf("Enter profits of the objects:\n");
    for(int i=0; i<n; i++){
        scanf("%d", &profits[i]);
    }

    int capacity;
    printf("Enter the capacity: ");
    scanf("%.d", &capacity);

    printf("\n objects:\n");
    printf("weight \t profit \n");
```

```
for (int i=0; i<n; i++){
    printf("%d\t%d\n", weight[i], profit[i]);
}
printf("\n Knapsack capacity : %d\n", capacity);

knapsack(n, capacity, weights, profits);

return 0;
}
```

Output :—

```
Enter number of objects: 4
Enter weights of the objects.
3, 7, 4, 8
Enter profits of the objects:
30, 50, 10, 80
Enter knapsack capacity : 15
objects:
weight    profit
3         80
7         50
4         10
8         80
Knapsack capacity 15
max profit : 100
Objects selected:
object 4 (weight = 8, profit = 80)
object 1 (weight = 3, profit = 30)
```