LAB-9

Question : 1. Implement 0/1 Knapsack problem using dynamic programming.

2. Find Minimum Cost Spanning Tree of a given undirected graph using Prims

1.SOURCE CODE:

```
#include <stdio.h>

#define MAX_OBJECTS 100
int max(int a, int b) {
  return (a > b) ? a : b;
}

void knapsack(int n, int W, int weights[], int profits[]) {
  int i, w;
  int K[MAX_OBJECTS + 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 (weights[i-1] <= w)
        K[i][w] = max(profits[i - 1] + K[i - 1][w - weights[i - 1]], K[i - 1][w]);
      else
        K[i][w] = K[i-1][w];
    }
  }

  for (i = 0; i <= n; i++) {
    for (w = 0; w <= W; w++) {
      printf("%d\t",K[i][w]);
    }
    printf("\n");
  }

  int maxProfit = K[n][W];
  printf("Maximum profit: %d\n", maxProfit);
```

```
  printf("Objects selected in the knapsack:\n");
  int res = maxProfit;
  w = W;
  for (i = n; i > 0 && res > 0; i--) {
    if (res == K[i - 1][w])
      continue;
    else {
      printf("Object %d (weight = %d, profit = %d)\n", i, weights[i - 1],
profits[i - 1]);
      res -= profits[i - 1];
      w -= weights[i - 1];
    }
  }
}

int main() {
  int n, W;
  int weights[MAX_OBJECTS], profits[MAX_OBJECTS];
  int i;
  printf("Enter number of objects (max %d): ", MAX_OBJECTS);
  scanf("%d", &n);
  printf("Enter the weights of the objects:\n");
  for (i = 0; i < n; i++) {
    scanf("%d", &weights[i]);
  }
  printf("Enter the profits of the objects:\n");
  for (i = 0; i < n; i++) {
    scanf("%d", &profits[i]);
  }
  printf("Enter the capacity of the knapsack: ");
  scanf("%d", &W);

  knapsack(n, W, weights, profits);

  return 0;
}
```

RESULT:

```
Enter number of objects (max 100): 4
Enter the weights of the objects:
2 1 3 2
Enter the profits of the objects:
12 10 20 15
Enter the capacity of the knapsack: 5
0       0       0       0       0       0
0       0       12      12      12      12
0       10      12      22      22      22
0       10      12      22      30      32
0       10      15      25      30      37
Maximum profit: 37
Objects selected in the knapsack:
Object 4 (weight = 2, profit = 15)
Object 2 (weight = 1, profit = 10)
Object 1 (weight = 2, profit = 12)

Process returned 0 (0x0)   execution time : 12.885 s
Press any key to continue.
```

2.SOURCE CODE:

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

#define MAX_VERTICES 100
#define INF INT_MAX // Infinity

int minKey(int n, int d[], int s[]) {
   int min = INF, min_index;

   for (int v = 0; v < n; v++) {
      if (s[v] == 0 && d[v] < min) {
         min = d[v];
         min_index = v;
      }
   }
```

```
    return min_index;
}

int printMST(int n, int p[], int cost[MAX_VERTICES][MAX_VERTICES]) {
    int total_cost = 0;
    printf("Edge   Weight\n");
    for (int i = 1; i < n; i++) {
        printf("%d - %d    %d \n", p[i], i, cost[i][p[i]]);
        total_cost += cost[i][p[i]];
    }
    return total_cost;
}

void primMST(int n, int cost[MAX_VERTICES][MAX_VERTICES]) {
    int p[MAX_VERTICES];
    int d[MAX_VERTICES];
    int s[MAX_VERTICES];

    for (int i = 0; i < n; i++) {
        d[i] = INF;
        s[i] = 0;
    }

    d[0] = 0;
    p[0] = -1;

    for (int count = 0; count < n - 1; count++) {
        int u = minKey(n, d, s);
        s[u] = 1;
        for (int v = 0; v < n; v++) {
            if (cost[u][v] && s[v] == 0 && cost[u][v] < d[v]) {
                p[v] = u;
                d[v] = cost[u][v];
            }
        }
    }
    int total_cost = printMST(n, p, cost);
    printf("Total cost of Minimum Spanning Tree (MST): %d\n", total_cost);
```

```
}

int main() {
   int n;
   int cost[MAX_VERTICES][MAX_VERTICES];
   printf("Enter number of vertices (max %d): ", MAX_VERTICES);
   scanf("%d", &n);
   printf("Enter the cost adjacency matrix (use %d for infinity):\n", INF);
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         scanf("%d", &cost[i][j]);
         if (cost[i][j] == 0 && i != j) {
            cost[i][j] = INF;
         }
      }
   }
   printf("Minimum Spanning Tree (MST) using Prim's algorithm:\n");
   primMST(n, cost);

   return 0;
}
```

RESULT:

```
Enter number of vertices (max 100): 5
Enter the cost adjacency matrix (use 2147483647 for infinity):
0 11 9 7 8
11 0 5 14 13
9 5 0 12 14
7 14 12 0 6
8 13 14 6 0
Minimum Spanning Tree (MST) using Prim's algorithm:
Edge   Weight
2 - 1    5
0 - 2    9
0 - 3    7
3 - 4    6
Total cost of Minimum Spanning Tree (MST): 27

Process returned 0 (0x0)   execution time : 26.406 s
Press any key to continue.
```