Lab - 9

①

Knapsack :-

```
#include <stdio.h>
# define N 4
# define CAPACITY 10

Struct Item {
        int weight;
        int profit;
  };

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

void Knapsack (Struct Item items [] , int n, int capacity ) {
     int dp[n+1] [capacity +1];
   for (int i= 0 ; i <= n ; i++) {
      for (int w=0 ; w< = capacity ; w++) {
          if (i== 0 || w==0)
            dp [i] [w] =0;
        else if (items [i-1].weight <= w)
            dp [i] [w]  = max (items  [i-1]. profit +dp [i-1] [w - items
                          [i-1]. weight], dp [i-1] [w]);
           else
             dp [i] [w]= dp [i- 1][w];
        }
   }
```

[illegible] 4/7/24

```
int remaining Capacity = Capacity;
printf ("Items Selected : \n");
for (int i = n ; i > 0 && max profit > 0 ; i--) {
    if (maxProfit != dp [i-1] [remaining Capacity ]) {
        printf ("Item %d (weight : %d, profit : %d) \n",
                i, items [i-1].weight, items [i-1]. profit);
        maxProfit -= items [i-1]. profit;
        remaining Capacity -= items [i-1].weight ;
      }
   }
}

int main () {
        struct Item items [N] = {
                          {2, 6},
                          {3, 5},
                          {4, 8},
                          {5, 9},
                   };
```

→ Output :- DP table

```
0   0   0   0   0   0
0   10  12  12  12  12
0   10  12  22  22  22
0   10  12  22  30  32
0   10  12  25  30  37
```

Max value = 37

Items included : Item 4 (Value : 15 , weight : 2)
Item 2 (value : 10 , weight : 1)
Item 1 (value : 12 , weight : 2)

② Prims :-

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

# define V 5

int minKey (int key [], bool mstSet [])
{
    int min = INT_MAX, min_index ;

    for (int v = 0; v < V; v++)
        if (mstSet [v] == false && key [v] < min)
            min = key [v], min_index = v;
    return min_index;
}

void printMST (int parent [], int graph [V][V])
{
    printf ("Edge \t weight \n");
     for (int i = 1 ; i < V; i++)
         print ("%d - %d \t %d \n", parent [i], i, graph [i]
                                   [parent [i]]);
}

void primMST (int graph [V][V])
{
    int parent [V];
    int key [V];
    bool mstSet [V];
```

[illegible] 4/3/24.

```
for (int i = 0; i < V; i++)
   Key [i] = INT_MAX,  mstSet[i] = false;
  Key [0] = 0;
  parent [0] = -1;

for (int count = 0; count < V-1; count ++) {
    int u = minKey (Key, mstSet);
    mst Set [u] = true;
   for (int v = 0;  v < V; v++)
       if  (graph [u] [v] && mstSet [v] == false
           graph [u] [v]  < Key[v])
         parent [v] = u,  Key [v] = graph [u] [v];
 }

int main ()
{
   int graph [V] [V] = {
                  {0, 2; 0, 6, 0},
                  { 2, 0, 3, 8, 5},
                  {0, 3, 0, 0, 7},
                  { 6, 8, 0, 0, 9},
                  {0, 5, 7, 9, 0},
          };
   primMST (graph);
   return 0;
}
```

→ Output :-

Edge	Weight
0 - 1	2
1 - 2	3
0 - 3	6
1 - 4	5