Lab-9

9/July/24

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

#define Max_Vertices 100
#define INF INT_MAX

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 Print MST (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 (int i = 1; i<n; i++) {
Printf ("%d - %d %d\n", P[i], i, cos[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 cost total_cost = PrintMST (n, P, cost);
Printf("\n Total cost of Minimum Spanning Tree(MST)
    : %d \n", totalcost);
}
int main() {
    int n;
    int cost [MAX_VERTICES] [MAX_VERTICES]
Printf ("Enter Number of Vertices (max %d ): ",
        MAX_VERTICES);
Scanf ("%d", &n);
Printf ("\n 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");
return 0;
```

Output:

Enter number of vertices (max 100): 4

Enter the cost adjacency matrix (use 2147483647 for infinity):

3 4 5 2147483647

9 0 2147483647 2

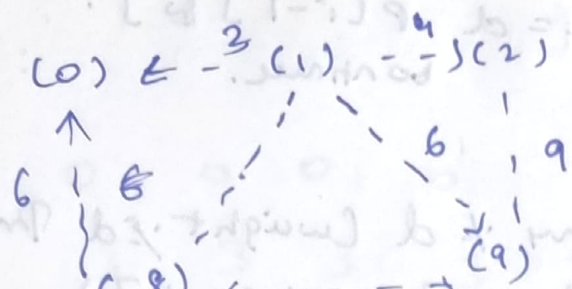
8 9 0 2147483647

2 9 4 5

Minimum Spanning Tree (MST) using Prim's algorithm:

Edge	Weight
0 - 1	9
3 - 2	2147483647
1 - 3	9

Total cost of Minimum Spanning Tree (MST): 60



4/7/24

Knapsack Problem
Dynamic Programming.

```
#include <stdio.h>
int max (int a, int b) {
    return (a > b) a : b
    }
void knapsack (int w, int wt[], int val[],
    int n) {
  int j, i, w;
  for (i=0; i<=n; i++) {
     for (w=0; w<=W; w++) {
        if (i==0 || w==0)
        dp[i][w]=0
```

Else if (wt[i-1] <= w)
dp[i][w] = max (val[i-1] + dp
[i-1][
[w - wt[i-1]], dp[i-1][w];
Else dp[i][w] = dp[i-1][w];
}
}

```
int res := dp[n][w]
print f ("minimum profit %d \n", res) }

w = W;
printf ("items included in the knapsack \n");
for (j=n; i>0 && res > 0; i--) {
```

```
if (res == dp[i-1][w])
    continue;

else {
    printf("Items %d Weight: %d Profit: %d \n",
        i, wt[i-1], val[i-1]);
    res = res - val[i-1];
    w = w - wt[i-1];
}
}
printf("\n DP Table : \n");
for (i=0; i<=n; i++) {
    for (w=0; w<=W; w++) {
        printf("%d ", dp[i][w]);
    }
    printf("\n");
}
}
int main() {
    int n = 4;
    int val[] = {3, 4, 5, 6};
    int wt[] = {2, 3, 4, 5};
    int W = 8;
    knapSack(W, wt, val, n);
    return 0;
}
```

Output

Max Profit = 10
Items included in knapsack;
Item 4 (weight: 5 Profit 6)
Item 2 (weight: 3, Profit 4)

DP table:

0	0	0	0	0	0	0	0	0
0	0	3	3	3	3	3	3	3
0	0	3	4	4	7	7	7	7
0	0	3	4	5	7	8	9	9
0	0	3	4	5	7	8	9	10