

4/7/24

Lab-9.

1. Prim's Algorithm

#include <stdio.h>

#include <limits.h>

#define N 4

int minKey(int Key[], int mstSet[]){

int min = INT_MAX, min-Index;

for (int v=0; v<N; v++)

if (mstSet[v] == 0 && key[v] < min)

min = key[v], min-index = v;

return min-index;

}

void printMST(int parent[], int graph[N][N]){

printf("Edge\tWeight\n");

for (int i=1; i<N; i++)

printf("%d - %d\t%d\n", parent[i], i, graph[i][parent[i]]);

}

void primMST(int graph[N][N]){

int parent[N];

int key[N];

int mstSet[N];

for (int i=0; i<N; i++)

key[i] = INT_MAX, mstSet[i] = 0;

key[0] = 0;

parent[0] = -1;

for (int count=0; count<N-1; count++){

int u = minKey(key, mstSet);

mstSet[u] = 1;

for (int v=0; v<N; v++){

if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])

parent[v] = u; key[v] = graph[u][v];

}


```

PrintMST (parent, graph);
}

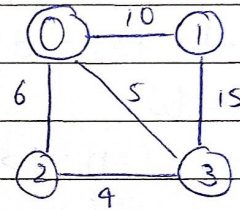
int main() {
    int graph[N][N] = { {0, 10, 6, 5}, {10, 0, 0, 15},
                        {6, 0, 0, 4}, {5, 15, 4, 0} };

    primMST (graph);
    return 0;
}

```

O/Ps -

Edge	Weight
0-1	10
3-2	4
0-3	5



2. Knapsack by Dynamic Programming.

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

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

```
{
```

```
void knapsack (int n, int w, int weights[], int values[])
```

```
    int i, w;
```

```
    int dp[n+1][w+1];
```

```
    for (i = 0; i <= n; i++) {
```

```
        for (w = 0; w <= W; w++) {
```

```
            if (i == 0 || w == 0) {
```

```
                dp[i][w] = 0;
```

```
            }
```

```
            else if (weights[i-1] <= w) {
```

```
                dp[i][w] = max (dp[i-1][w], dp[i-1][w -
                                weights[i-1] + values[i-1]);
```

```
            }
```

```
        else {
```

```
            dp[i][w] = dp[i-1][w];
```

```
        }
```

```

    }
}
printf("DP Table: \n");
for (i = 0; i <= n; i++) {
    for (w = 0; w <= W; w++) {
        printf("%d\t", dp[i][w]);
    }
    printf("\n");
}
printf("Selected items:");
int res = dp[n][W];
w = W;
for (i = n; i > 0 && res > 0; i--) {
    if (res == dp[i-1][w])
        continue;
    else {
        printf("%d", i);
        res = res - values[i-1];
        w = w - weights[i-1];
    }
}
printf("\n Maximum profit: %d\n", dp[n][W]);
}

```

O/P:-

DP Table:

0	0	0	0	0	0
0	0	3	3	3	3
0	0	3	4	4	7
0	0	3	4	5	7
0	0	3	4	5	8

Selected items = 4

Maximum profit = 8.

Q
4/7/24