4-7-24

Lab - 9

~~① Perform knapsack problem using dynamic programming~~

① Perform the prim's algorithm.

```
#include <stdio.h>
#include <stdbool.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++)
        printf ("%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];
    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);
        mstSet[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];
    }
```

```
    printMST(parent, graph);
}

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} };
    printMST(graph);
    return 0;
}
```

Output:

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

② Perform knapsack problem using 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 i, w;
    int K[n+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(wt[i-1] <= w) {
                K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);
            }
```

4/7/24

```
            else {
                K[i][w] = K[i-1][w];
            }
        }
    }

    printf("DP table:\n");
    for (i=0; i<=n; i++) {
        for (w=0; w<=W; w++) {
            printf("%4d", K[i][w]);
        }
        printf("\n");
    }

    int res = K[n][W];
    printf("\nMaximum value in Knapsack = %d\n", res);
    printf("Items included in the knapsack:\n");
    w = W;
    for (i=n; i>0 && res>0; i--) {
        if (res == K[i-1][w])
            continue;
        else {
            printf("Item %d (value: %d, weight: %d)\n", i, val[i-1], wt[i-1]);
            res -= val[i-1];
            w -= wt[i-1];
        }
    }
}

int main() {
    int n = 4;
    int val[] = {12, 10, 20, 15};
    int wt[] = {2, 1, 3, 2};
    int W = 5;
    knapsack(W, wt, val, n);
    return 0;
}
```

Output:-

DP Table:

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 value in knapsack = 37

Items included in the knapsack :

Item 4 (Value: 15, weight: 2)

Item 2 (Value: 10, weight: 1)

Item 1 (Value: 12, Weight : 2)

4/7/24