LAB-9.

Date
Page 04/07/24

① Perform knapsack problem using DP:

```
#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];
        }
    }

    int maxProfit = dp[n][capacity];
    printf("Maximum Profit : %d\n", maxProfit);
```

4/7/24

BU-3

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

int main() {
    struct Items items[n] = {
        {2,6},
    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;
}
```

o/p:

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 = 37

items included in the knapsack:

items 4, item 2, item 1

(2) Prims 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("Edges \t Weight \n");
    for(int i= 1; i< V ; i++) {
        printf("%d - %d \t %d \n", parent[i]-1, graph[i][parent[i]];
    }
}

void primMST(int graph[v][v]){
    int parent[v];
    int key[v];
    int mstKey[v];
    for(int i=0; i<V; i++)
        key[i] = INT_MAX, mstSet[i] = false;
    Key[0] = 0;
    Parent[0] = -1;
```

4/7/24

```
for(int count=0; count < V ; 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);
return 0;
}
```

o/p:

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