Date: 10.05.2024 Name: Mugilkrishna D U
Ex. No: 9 Reg. No.: 3122225001073

UCS 2403 Design & Analysis of Algorithms

Assignment 9

Date of Exercise: 10.05.2024

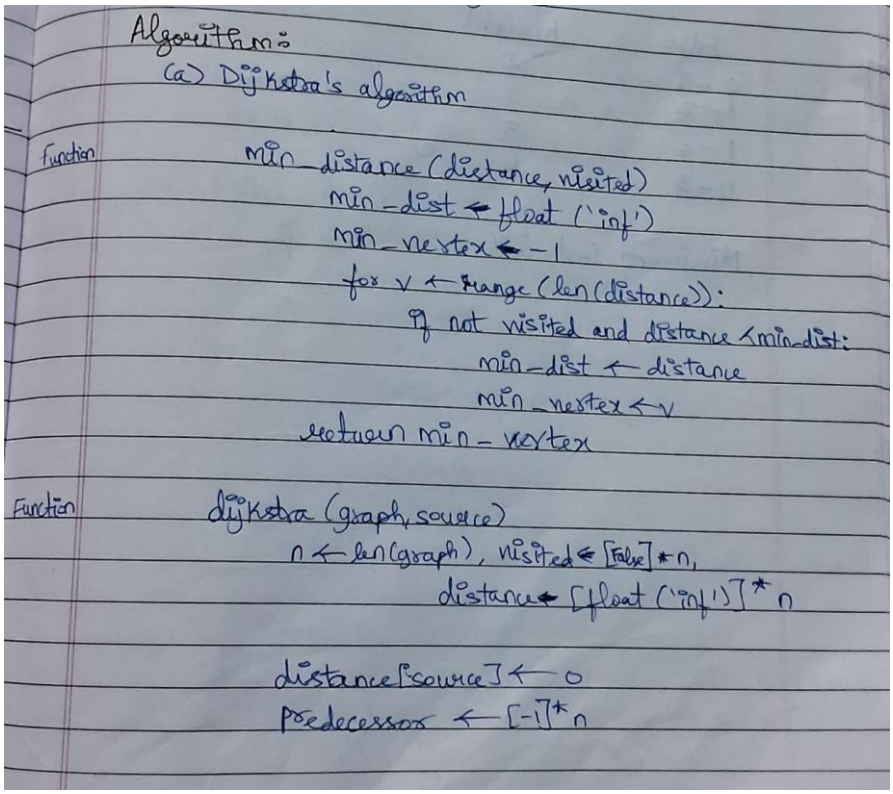
Aim: To gain understanding and proficiency on solving problems using Greedy Algorithms

Question 1:

Using adjacency matrix representation for the input graph, implement the following algorithms:

- Dijkstra's algorithm
- Prim's algorithm
- Kruskal's algorithm

Algorithms:



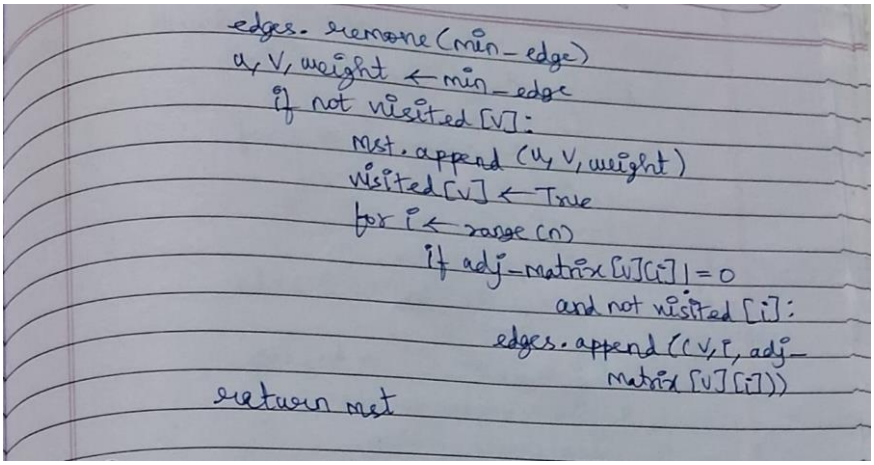
Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
for i ← range(n)
    u ← min_distance(distance, visited)
    visited[u] = True

for v ← range(n):
    if graph[u][v] != 0 and not visited[v]:
        new_dist ← distance[u] +
                   graph[u][v]
        if new_dist < distance[v]:
            distance[v] ← new_dist
            predecessor[v] = u

return distance, predecessor
```

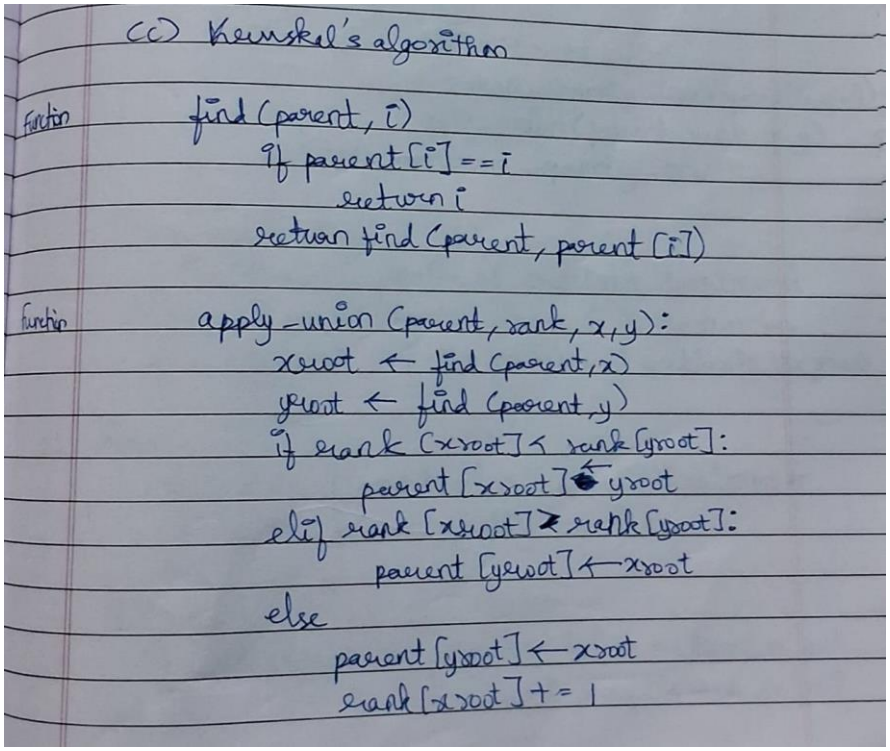
(b) Prim's algorithm:

```
Function
prims(adj_matrix)
    n ← len(adj_matrix)
    mst ← []
    visited ← [False] * n
    visited[0] ← True, edges ← []

    for i ← range(1,n)
        if adj_matrix[0][i] != 0:
            edges.append(0, i, adj_matrix[0]
                         [i])

    while len(mst) < n-1:
        min_edge ← min(edges, key =
                   lambda x: x[2])
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

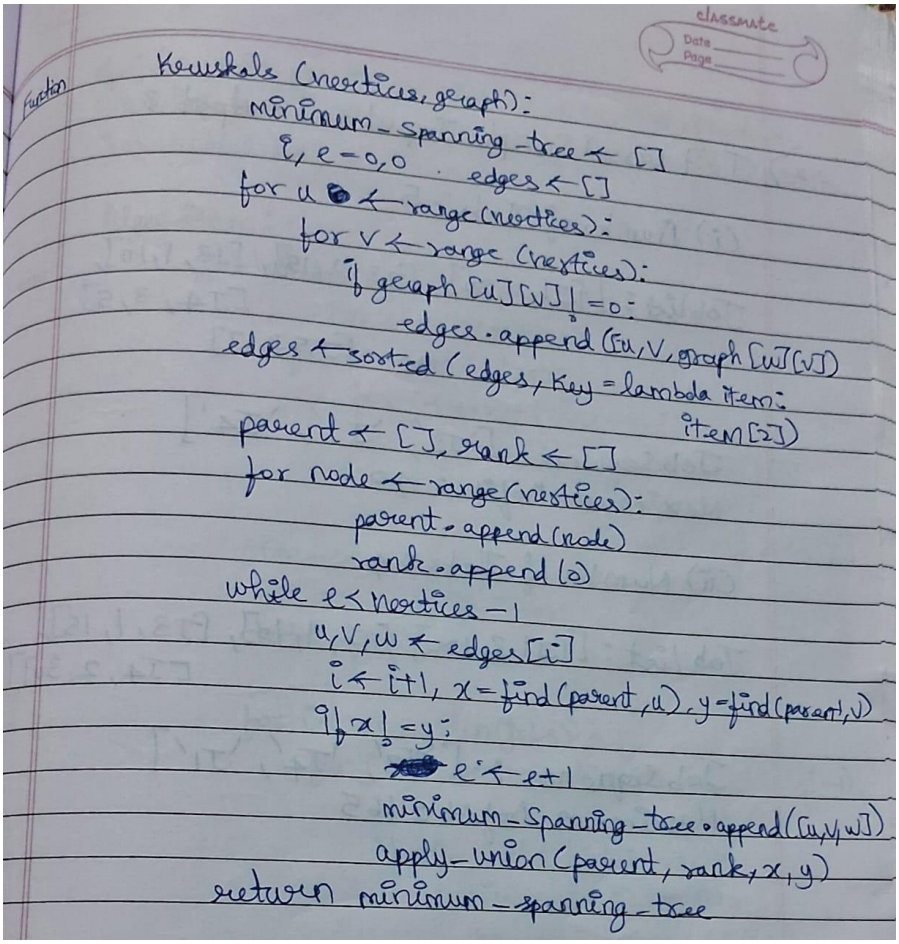


```
edges.remove(min_edge)
u, v, weight ← min_edge
if not visited[v]:
    mst.append((u, v, weight))
    visited[v] ← True
    for i ← range(n)
        if adj_matrix[v][i] != 0
                and not visited[i]:
            edges.append((v, i, adj_
                matrix[v][i]))
return mst
```



```
(c) Kruskal's algorithm

function find(parent, i)
    if parent[i] == i
        return i
    return find(parent, parent[i])

function apply_union(parent, rank, x, y):
    xroot ← find(parent, x)
    yroot ← find(parent, y)
    if rank[xroot] < rank[yroot]:
        parent[xroot] ← yroot
    elif rank[xroot] > rank[yroot]:
        parent[yroot] ← xroot
    else
        parent[yroot] ← xroot
        rank[xroot] += 1
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073



CODE:

Code for Dijkstra's Algorithm:

```
def min_distance(distance, visited):
    min_dist = float('inf')
    min_vertex = -1
    for v in range(len(distance)):
        if not visited[v] and distance[v] < min_dist:
            min_dist = distance[v]
            min_vertex = v
    return min_vertex

def dijkstra(graph, source):
    n = len(graph)
    visited = [False] * n
    distance = [float('inf')] * n

    distance[source] = 0
    predecessor = [-1] * n
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
    for _ in range(n):
        u = min_distance(distance, visited)
        visited[u] = True

        for v in range(n):
            if graph[u][v] != 0 and not visited[v]:
                new_dist = distance[u] + graph[u][v]

                if new_dist < distance[v]:
                    distance[v] = new_dist
                    predecessor[v] = u

    return distance, predecessor

def printSolution(distance, predecessor, source):
    print("Vertex \t Shortest Distance from Source \t\t Path")
    for i in range(len(distance)):
        if distance[i] == float('inf'):
            print(f"{i} \t\t\t {distance[i]} \t\t\t No path")
        else:
            print(f"{i} \t\t\t {distance[i]} \t\t\t {get_path(predecessor, i,
source)}")

def get_path(predecessor, vertex, source):
    path = []
    current = vertex
    while current != -1:
        path.append(current)
        current = predecessor[current]
    path.reverse()
    return ' -> '.join(map(str, path))

vertices = int(input("Enter the number of vertices: "))
source_vertex = int(input("Enter the source vertex: "))

print("\n")
adj_matrix = []
for i in range(vertices):
    row = []
    for j in range(vertices):
        element = int(input(f"Enter the weight from vertex {i} to vertex {j}: "))
        row.append(element)
    print("\n")
    adj_matrix.append(row)

shortest_distance, predecessor = dijkstra(adj_matrix, source_vertex)
printSolution(shortest_distance, predecessor, source_vertex)
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.1.1.py"
Enter the number of vertices: 9
Enter the source vertex: 0


Enter the weight from vertex 0 to vertex 0: 0
Enter the weight from vertex 0 to vertex 1: 4
Enter the weight from vertex 0 to vertex 2: 0
Enter the weight from vertex 0 to vertex 3: 0
Enter the weight from vertex 0 to vertex 4: 0
Enter the weight from vertex 0 to vertex 5: 0
Enter the weight from vertex 0 to vertex 6: 0
Enter the weight from vertex 0 to vertex 7: 8
Enter the weight from vertex 0 to vertex 8: 0


Enter the weight from vertex 1 to vertex 0: 4
Enter the weight from vertex 1 to vertex 1: 0
Enter the weight from vertex 1 to vertex 2: 8
Enter the weight from vertex 1 to vertex 3: 0
Enter the weight from vertex 1 to vertex 4: 0
Enter the weight from vertex 1 to vertex 5: 0
Enter the weight from vertex 1 to vertex 6: 0
Enter the weight from vertex 1 to vertex 7: 11
Enter the weight from vertex 1 to vertex 8: 0


Enter the weight from vertex 2 to vertex 0: 0
Enter the weight from vertex 2 to vertex 1: 8
Enter the weight from vertex 2 to vertex 2: 0
Enter the weight from vertex 2 to vertex 3: 7
Enter the weight from vertex 2 to vertex 4: 0
Enter the weight from vertex 2 to vertex 5: 4
Enter the weight from vertex 2 to vertex 6: 0
Enter the weight from vertex 2 to vertex 7: 0
Enter the weight from vertex 2 to vertex 8: 2


Enter the weight from vertex 3 to vertex 0: 0
Enter the weight from vertex 3 to vertex 1: 0
Enter the weight from vertex 3 to vertex 2: 7
Enter the weight from vertex 3 to vertex 3: 0
Enter the weight from vertex 3 to vertex 4: 9
Enter the weight from vertex 3 to vertex 5: 14
Enter the weight from vertex 3 to vertex 6: 0
Enter the weight from vertex 3 to vertex 7: 0
Enter the weight from vertex 3 to vertex 8: 0


Enter the weight from vertex 4 to vertex 0: 0
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
Enter the weight from vertex 4 to vertex 1: 0
Enter the weight from vertex 4 to vertex 2: 0
Enter the weight from vertex 4 to vertex 3: 9
Enter the weight from vertex 4 to vertex 4: 0
Enter the weight from vertex 4 to vertex 5: 10
Enter the weight from vertex 4 to vertex 6: 0
Enter the weight from vertex 4 to vertex 7: 0
Enter the weight from vertex 4 to vertex 8: 0


Enter the weight from vertex 5 to vertex 0: 0
Enter the weight from vertex 5 to vertex 1: 0
Enter the weight from vertex 5 to vertex 2: 4
Enter the weight from vertex 5 to vertex 3: 14
Enter the weight from vertex 5 to vertex 4: 10
Enter the weight from vertex 5 to vertex 5: 0
Enter the weight from vertex 5 to vertex 6: 2
Enter the weight from vertex 5 to vertex 7: 0
Enter the weight from vertex 5 to vertex 8: 0


Enter the weight from vertex 6 to vertex 0: 0
Enter the weight from vertex 6 to vertex 1: 0
Enter the weight from vertex 6 to vertex 2: 0
Enter the weight from vertex 6 to vertex 3: 0
Enter the weight from vertex 6 to vertex 4: 0
Enter the weight from vertex 6 to vertex 5: 2
Enter the weight from vertex 6 to vertex 6: 0
Enter the weight from vertex 6 to vertex 7: 1
Enter the weight from vertex 6 to vertex 8: 6


Enter the weight from vertex 7 to vertex 0: 8
Enter the weight from vertex 7 to vertex 1: 11
Enter the weight from vertex 7 to vertex 2: 0
Enter the weight from vertex 7 to vertex 3: 0
Enter the weight from vertex 7 to vertex 4: 0
Enter the weight from vertex 7 to vertex 5: 0
Enter the weight from vertex 7 to vertex 6: 1
Enter the weight from vertex 7 to vertex 7: 0
Enter the weight from vertex 7 to vertex 8: 7


Enter the weight from vertex 8 to vertex 0: 0
Enter the weight from vertex 8 to vertex 1: 0
Enter the weight from vertex 8 to vertex 2: 2
Enter the weight from vertex 8 to vertex 3: 0
Enter the weight from vertex 8 to vertex 4: 0
Enter the weight from vertex 8 to vertex 5: 0
Enter the weight from vertex 8 to vertex 6: 6
Enter the weight from vertex 8 to vertex 7: 7
Enter the weight from vertex 8 to vertex 8: 0
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
Vertex   Shortest Distance from Source          Path
0                       0                      0
1                       4                      0 -> 1
2                       12                     0 -> 1 -> 2
3                       19                     0 -> 1 -> 2 -> 3
4                       21                     0 -> 7 -> 6 -> 5 -> 4
5                       11                     0 -> 7 -> 6 -> 5
6                       9                      0 -> 7 -> 6
7                       8                      0 -> 7
8                       14                     0 -> 1 -> 2 -> 8
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity of Dijkstra's algorithm is O(V^2) in the above code implementation but can be reduced to O(V + E logV) by using min-priority queue

Code for Prim's Algorithm:

```python
def prims(adj_matrix):
    n = len(adj_matrix)
    mst = []
    visited = [False] * n
    visited[0] = True
    edges = []

    for i in range(1, n):
        if adj_matrix[0][i] != 0:
            edges.append((0, i, adj_matrix[0][i]))

    while len(mst) < n - 1:
        min_edge = min(edges, key=lambda x: x[2])
        edges.remove(min_edge)
        u, v, weight = min_edge
        if not visited[v]:
            mst.append((u, v, weight))
            visited[v] = True
            for i in range(n):
                if adj_matrix[v][i] != 0 and not visited[i]:
                    edges.append((v, i, adj_matrix[v][i]))
    return mst

vertices = int(input("Enter the number of vertices: "))
print("\n")

adj_matrix = []
for i in range(vertices):
    row = []
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
    for j in range(vertices):
        element = int(input(f"Enter the weight from vertex {i} to vertex {j}: "))
        row.append(element)
    print("\n")
    adj_matrix.append(row)

minimum_spanning_tree = prims(adj_matrix)
print("-----MINIMUM SPANNING TREE-----")
print("Edge \t Weight")
total_cost = 0
for u, v, weight in minimum_spanning_tree:
    print(f"{u} - {v} \t {weight}")
    total_cost += weight

print("\nThe Minimum cost is: ", total_cost)
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/9.1.2.py"
Enter the number of vertices: 5


Enter the weight from vertex 0 to vertex 0: 0
Enter the weight from vertex 0 to vertex 1: 2
Enter the weight from vertex 0 to vertex 2: 0
Enter the weight from vertex 0 to vertex 3: 6
Enter the weight from vertex 0 to vertex 4: 0


Enter the weight from vertex 1 to vertex 0: 2
Enter the weight from vertex 1 to vertex 1: 0
Enter the weight from vertex 1 to vertex 2: 3
Enter the weight from vertex 1 to vertex 3: 8
Enter the weight from vertex 1 to vertex 4: 5


Enter the weight from vertex 2 to vertex 0: 0
Enter the weight from vertex 2 to vertex 1: 3
Enter the weight from vertex 2 to vertex 2: 0
Enter the weight from vertex 2 to vertex 3: 0
Enter the weight from vertex 2 to vertex 4: 7
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
Enter the weight from vertex 3 to vertex 0: 6
Enter the weight from vertex 3 to vertex 1: 8
Enter the weight from vertex 3 to vertex 2: 0
Enter the weight from vertex 3 to vertex 3: 0
Enter the weight from vertex 3 to vertex 4: 9


Enter the weight from vertex 4 to vertex 0: 0
Enter the weight from vertex 4 to vertex 1: 5
Enter the weight from vertex 4 to vertex 2: 7
Enter the weight from vertex 4 to vertex 3: 9
Enter the weight from vertex 4 to vertex 4: 0


-----MINIMUM SPANNING TREE-----
Edge     Weight
0 - 1    2
1 - 2    3
1 - 4    5
0 - 3    6

The Minimum cost is:  16
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity of Prim's algorithm is O(V^2) in the above code implementation but can be reduced to O(V+E logV) by using a priority queue

Code for Kruskal's Algorithm:

```python
def find(parent, i):
    if parent[i] == i:
        return i
    return find(parent, parent[i])

def apply_union(parent, rank, x, y):
    xroot = find(parent, x)
    yroot = find(parent, y)
    if rank[xroot] < rank[yroot]:
        parent[xroot] = yroot
    elif rank[xroot] > rank[yroot]:
        parent[yroot] = xroot
    else:
        parent[yroot] = xroot
        rank[xroot] += 1

def kruskals(vertices, graph):
    minimum_spanning_tree = []
    i, e = 0, 0
    edges = []
    for u in range(vertices):
        for v in range(vertices):
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
            if graph[u][v] != 0:
                edges.append([u, v, graph[u][v]])
    edges = sorted(edges, key=lambda item: item[2])
    parent = []
    rank = []
    for node in range(vertices):
        parent.append(node)
        rank.append(0)
    while e < vertices - 1:
        u, v, w = edges[i]
        i = i + 1
        x = find(parent, u)
        y = find(parent, v)
        if x != y:
            e = e + 1
            minimum_spanning_tree.append([u, v, w])
            apply_union(parent, rank, x, y)

    return minimum_spanning_tree

vertices = int(input("Enter the number of vertices: "))
print("\n")

adj_matrix = []
for i in range(vertices):
    row = []
    for j in range(vertices):
        element = int(input(f"Enter the weight from vertex {i} to vertex {j}: "))
        row.append(element)
    print("\n")
    adj_matrix.append(row)

minimum_spanning_tree = kruskals(vertices, adj_matrix)

print("-----MINIMUM SPANNING TREE-----")
print("Edge \t Weight")
total_cost = 0
for u, v, weight in minimum_spanning_tree:
    print(f"{u} - {v} \t {weight}")
    total_cost += weight

print("\nThe Minimum cost is: ", total_cost)
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.1.3.py"
Enter the number of vertices: 9


Enter the weight from vertex 0 to vertex 0: 0
Enter the weight from vertex 0 to vertex 1: 4
Enter the weight from vertex 0 to vertex 2: 0
Enter the weight from vertex 0 to vertex 3: 0
Enter the weight from vertex 0 to vertex 4: 0
Enter the weight from vertex 0 to vertex 5: 0
Enter the weight from vertex 0 to vertex 6: 0
Enter the weight from vertex 0 to vertex 7: 8
Enter the weight from vertex 0 to vertex 8: 0


Enter the weight from vertex 1 to vertex 0: 4
Enter the weight from vertex 1 to vertex 1: 0
Enter the weight from vertex 1 to vertex 2: 8
Enter the weight from vertex 1 to vertex 3: 0
Enter the weight from vertex 1 to vertex 4: 0
Enter the weight from vertex 1 to vertex 5: 0
Enter the weight from vertex 1 to vertex 6: 0
Enter the weight from vertex 1 to vertex 7: 11
Enter the weight from vertex 1 to vertex 8: 0


Enter the weight from vertex 2 to vertex 0: 0
Enter the weight from vertex 2 to vertex 1: 8
Enter the weight from vertex 2 to vertex 2: 0
Enter the weight from vertex 2 to vertex 3: 7
Enter the weight from vertex 2 to vertex 4: 0
Enter the weight from vertex 2 to vertex 5: 4
Enter the weight from vertex 2 to vertex 6: 0
Enter the weight from vertex 2 to vertex 7: 0
Enter the weight from vertex 2 to vertex 8: 2


Enter the weight from vertex 3 to vertex 0: 0
Enter the weight from vertex 3 to vertex 1: 0
Enter the weight from vertex 3 to vertex 2: 7
Enter the weight from vertex 3 to vertex 3: 0
Enter the weight from vertex 3 to vertex 4: 9
Enter the weight from vertex 3 to vertex 5: 14
Enter the weight from vertex 3 to vertex 6: 0
Enter the weight from vertex 3 to vertex 7: 0
Enter the weight from vertex 3 to vertex 8: 0


Enter the weight from vertex 4 to vertex 0: 0
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
Enter the weight from vertex 4 to vertex 1: 0
Enter the weight from vertex 4 to vertex 2: 0
Enter the weight from vertex 4 to vertex 3: 9
Enter the weight from vertex 4 to vertex 4: 0
Enter the weight from vertex 4 to vertex 5: 10
Enter the weight from vertex 4 to vertex 6: 0
Enter the weight from vertex 4 to vertex 7: 0
Enter the weight from vertex 4 to vertex 8: 0


Enter the weight from vertex 5 to vertex 0: 0
Enter the weight from vertex 5 to vertex 1: 0
Enter the weight from vertex 5 to vertex 2: 4
Enter the weight from vertex 5 to vertex 3: 14
Enter the weight from vertex 5 to vertex 4: 10
Enter the weight from vertex 5 to vertex 5: 0
Enter the weight from vertex 5 to vertex 6: 2
Enter the weight from vertex 5 to vertex 7: 0
Enter the weight from vertex 5 to vertex 8: 0


Enter the weight from vertex 6 to vertex 0: 0
Enter the weight from vertex 6 to vertex 1: 0
Enter the weight from vertex 6 to vertex 2: 0
Enter the weight from vertex 6 to vertex 3: 0
Enter the weight from vertex 6 to vertex 4: 0
Enter the weight from vertex 6 to vertex 5: 2
Enter the weight from vertex 6 to vertex 6: 0
Enter the weight from vertex 6 to vertex 7: 1
Enter the weight from vertex 6 to vertex 8: 6


Enter the weight from vertex 7 to vertex 0: 8
Enter the weight from vertex 7 to vertex 1: 11
Enter the weight from vertex 7 to vertex 2: 0
Enter the weight from vertex 7 to vertex 3: 0
Enter the weight from vertex 7 to vertex 4: 0
Enter the weight from vertex 7 to vertex 5: 0
Enter the weight from vertex 7 to vertex 6: 1
Enter the weight from vertex 7 to vertex 7: 0
Enter the weight from vertex 7 to vertex 8: 7


Enter the weight from vertex 8 to vertex 0: 0
Enter the weight from vertex 8 to vertex 1: 0
Enter the weight from vertex 8 to vertex 2: 2
Enter the weight from vertex 8 to vertex 3: 0
Enter the weight from vertex 8 to vertex 4: 0
Enter the weight from vertex 8 to vertex 5: 0
Enter the weight from vertex 8 to vertex 6: 6
Enter the weight from vertex 8 to vertex 7: 7
Enter the weight from vertex 8 to vertex 8: 0
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
-----MINIMUM SPANNING TREE-----
Edge     Weight
6 - 7    1
2 - 8    2
5 - 6    2
0 - 1    4
2 - 5    4
2 - 3    7
0 - 7    8
3 - 4    9

The Minimum cost is:  37
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

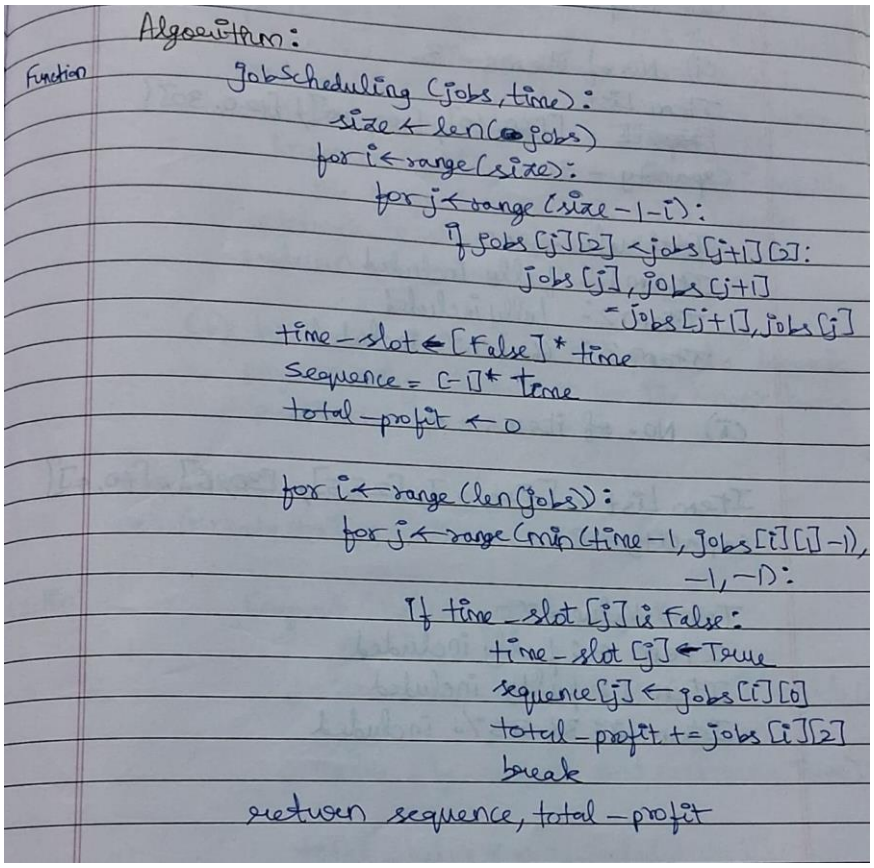
Time Complexity:

The Time Complexity of Kruskal's algorithm is O(E logE) or O(E logV), where E is the number of edges and V is the number of vertices in the graph

Question 2:

The job scheduling problem takes as input a set of jobs with deadlines and profits. A subset of jobs with maximum profit is the final output. Develop and implement a greedy algorithm to solve the job scheduling problem.

Algorithm:



Name: Mugilkrishna D U
Reg. No.: 3122225001073

Code:

```
def jobScheduling(jobs, time):
    size = len(jobs)
    for i in range(size):
        for j in range(size - 1 - i):
            if jobs[j][2] < jobs[j + 1][2]:
                jobs[j], jobs[j + 1], = jobs[j + 1], jobs[j]

    time_slot = [False] * time
    sequence = ['-1'] * time
    total_profit = 0

    for i in range(len(jobs)):
        for j in range(min(time - 1, jobs[i][1] - 1), -1, -1):
            if time_slot[j] is False:
                time_slot[j] = True
                sequence[j] = jobs[i][0]
                total_profit += jobs[i][2]
                break
    return sequence, total_profit
numberOfJobs = int(input("Enter the number of jobs: "))
print("\n")
jobs = []
deadline_list = []
for i in range(numberOfJobs):
    job = []
    jobName = str(input("Enter the job name: "))
    deadline = int(input("Enter the deadline: "))
    deadline_list.append(deadline)
    profit = int(input("Enter the profit: "))
    job.append(jobName)
    job.append(deadline)
    job.append(profit)
    jobs.append(job)
    print("\n")

timeLimit = max(deadline_list)
resultSequence, maxProfit = jobScheduling(jobs, timeLimit)
print("JOB SEQUENCE: ", resultSequence)
print("MAXIMUM PROFIT: ", maxProfit)
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.2.py"
Enter the number of jobs: 5


Enter the job name: J1
Enter the deadline: 2
Enter the profit: 20


Enter the job name: J2
Enter the deadline: 2
Enter the profit: 15


Enter the job name: J3
Enter the deadline: 1
Enter the profit: 10


Enter the job name: J4
Enter the deadline: 3
Enter the profit: 5
```

```
Enter the job name: J5
Enter the deadline: 3
Enter the profit: 1


JOB SEQUENCE:  ['J2', 'J1', 'J4']
MAXIMUM PROFIT:  40
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.2.py"
Enter the number of jobs: 4


Enter the job name: J1
Enter the deadline: 3
Enter the profit: 120


Enter the job name: J2
Enter the deadline: 1
Enter the profit: 10


Enter the job name: J3
Enter the deadline: 1
Enter the profit: 15
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
Enter the job name: J4
Enter the deadline: 2
Enter the profit: 30


JOB SEQUENCE:  ['J3', 'J4', 'J1']
MAXIMUM PROFIT:  165
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity for the job scheduling algorithm is O(n^2)

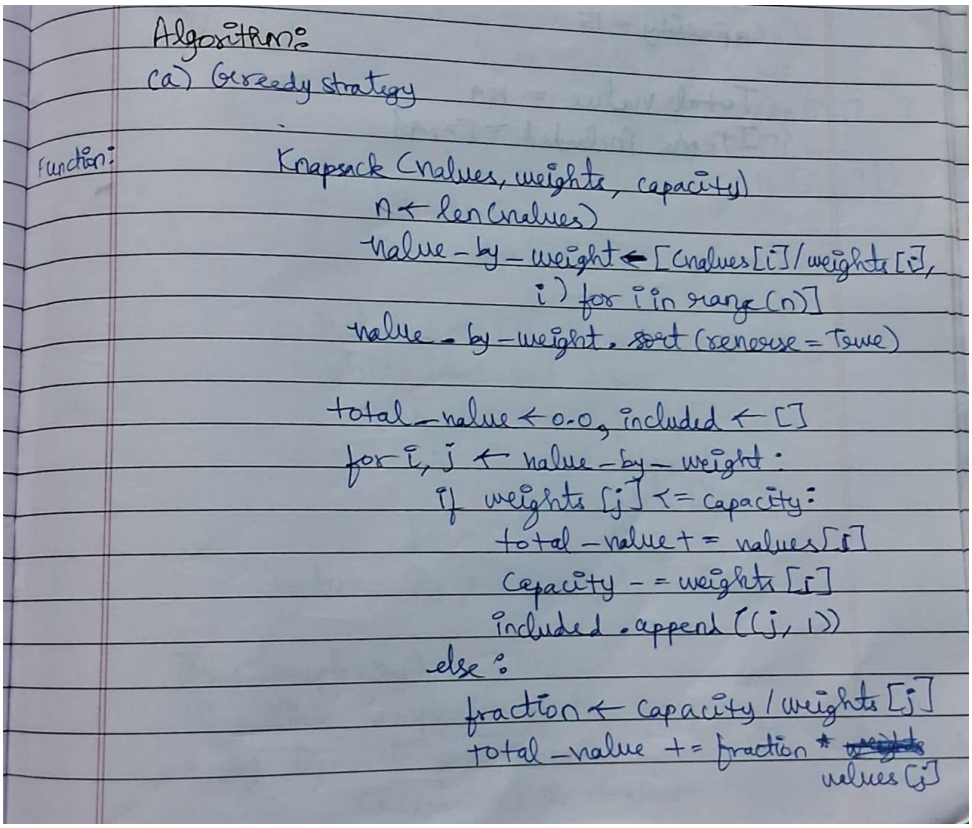
Question 3:

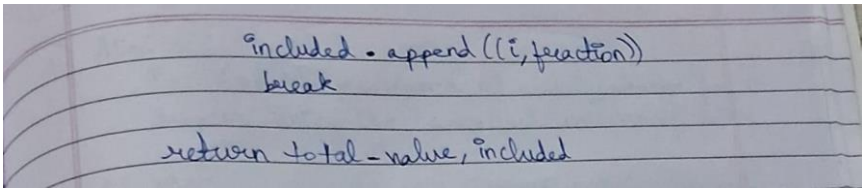
Consider the knapsack problem studied in the class.

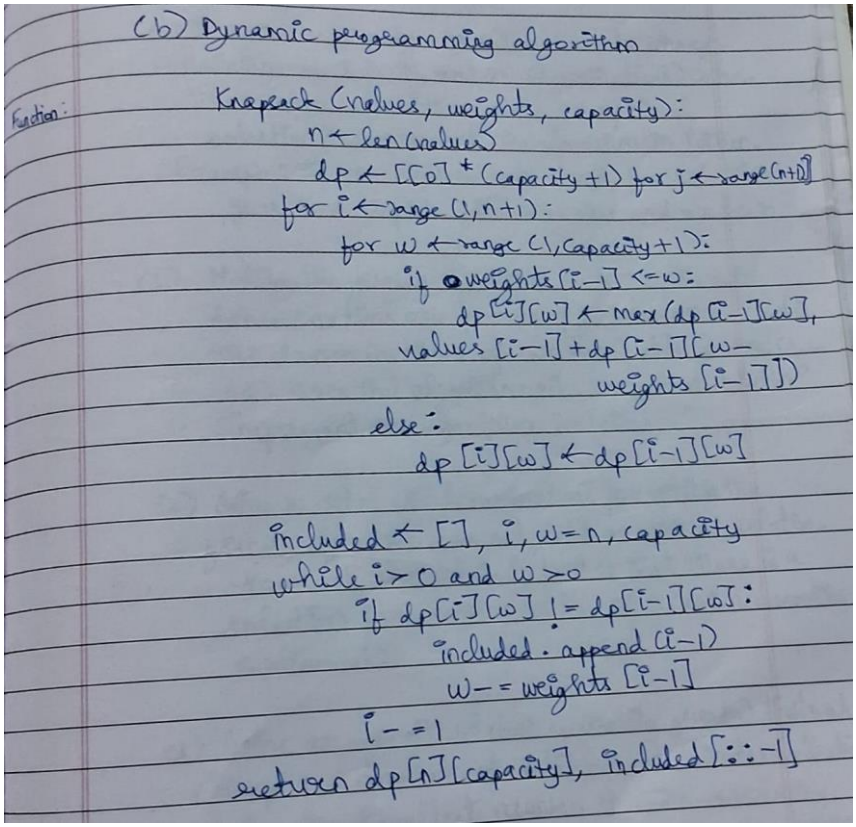
(a) Write a Python code to implement the following greedy strategy. Pick the item that has the maximum price per unit weight and add to the knapsack. Continue adding items until no more items can be added to the knapsack (adding any more results in exceeding the knapsack capacity). Give a counterexample (different from the one taught in the class) to show that this strategy may not return an optimal solution all the time.

(b) Write the Python code to implement a dynamic programming algorithm for the knapsack problem.

Algorithm:







CODE:

Code for Fractional Knapsack problem using greedy strategy:

```
def knapsack(values, weights, capacity):
    n = len(values)
    value_by_weight = [(values[i] / weights[i], i) for i in range(n)]
    value_by_weight.sort(reverse=True)

    total_value = 0.0
    included = []

    for _, i in value_by_weight:
        if weights[i] <= capacity:
            total_value += values[i]
            capacity -= weights[i]
            included.append((i, 1))
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
        else:
            fraction = capacity / weights[i]
            total_value += fraction * values[i]
            included.append((i, fraction))
            break

    return total_value, included

items = int(input("Enter the number of items: "))
print("\n")

values = []
weights = []

for i in range(items):
    value_item = int(input("Enter the value of item " + str(i + 1) + ": "))
    values.append(value_item)
    weight_item = int(input("Enter the weight of item " + str(i + 1) + ": "))
    weights.append(weight_item)
    print("\n")

capacity = int(input("Enter the capacity of the knapsack: "))

total_value, included_items = knapsack(values, weights, capacity)
print("\nTotal value:", total_value)
print("Items included:")
for item, fraction in included_items:
    if fraction == 1:
        print(f"Item {item + 1}: fully included")
    else:
        print(f"Item {item + 1}: {fraction * 100:.2f}% included")
```

Output:

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.3.1.py"
Enter the number of items: 3


Enter the value of item 1: 60
Enter the weight of item 1: 10


Enter the value of item 2: 100
Enter the weight of item 2: 20


Enter the value of item 3: 120
Enter the weight of item 3: 30


Enter the capacity of the knapsack: 50

Total value: 240.0
Items included:
Item 1: fully included
Item 2: fully included
Item 3: 66.67% included
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.3.1.py"
Enter the number of items: 4


Enter the value of item 1: 20
Enter the weight of item 1: 2


Enter the value of item 2: 25
Enter the weight of item 2: 5


Enter the value of item 3: 30
Enter the weight of item 3: 6


Enter the value of item 4: 40
Enter the weight of item 4: 4


Enter the capacity of the knapsack: 8

Total value: 70.0
Items included:
Item 4: fully included
Item 1: fully included
Item 3: 33.33% included
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

Time Complexity:

The Time Complexity of the Knapsack problem using the Greedy strategy is O(n logn)

Code for Knapsack problem using dynamic programming algorithm:

```
def knapsack(values, weights, capacity):
    n = len(values)
    dp = [[0] * (capacity + 1) for j in range(n + 1)]

    for i in range(1, n + 1):
        for w in range(1, capacity + 1):
            if weights[i - 1] <= w:
                dp[i][w] = max(dp[i - 1][w], values[i - 1] + dp[i - 1][w -
weights[i - 1]])
            else:
                dp[i][w] = dp[i - 1][w]

    included = []
    i, w = n, capacity
    while i > 0 and w > 0:
        if dp[i][w] != dp[i - 1][w]:
            included.append(i - 1)
            w -= weights[i - 1]
        i -= 1

    return dp[n][capacity], included[::-1]

items = int(input("Enter the number of items: "))
print("\n")

values = []
weights = []

for i in range(items):
    value_item = int(input("Enter the value of item " + str(i+1) + ": "))
    values.append(value_item)
    weight_item = int(input("Enter the weight of item " + str(i+1) + ": "))
    weights.append(weight_item)
    print("\n")

capacity = int(input("Enter the capacity of the knapsack: "))

total_value, included_items = knapsack(values, weights, capacity)
print("\nTotal value:", total_value)
print("Items included:", [i + 1 for i in included_items])
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.3.2.py"
Enter the number of items: 3


Enter the value of item 1: 60
Enter the weight of item 1: 10


Enter the value of item 2: 100
Enter the weight of item 2: 20


Enter the value of item 3: 120
Enter the weight of item 3: 30


Enter the capacity of the knapsack: 50

Total value: 220
Items included: [2, 3]
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.3.2.py"
Enter the number of items: 4
Enter the value of item 1: 30
Enter the weight of item 1: 3


Enter the value of item 2: 45
Enter the weight of item 2: 5


Enter the value of item 3: 50
Enter the weight of item 3: 10


Enter the value of item 4: 64
Enter the weight of item 4: 8


Enter the capacity of the knapsack: 15

Total value: 109
Items included: [2, 4]
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity of the Knapsack problem using Dynamic Programming approach is O(n * capacity) where the capacity is the maximum weight the Knapsack can hold

Name: Mugilkrishna D U
Reg. No.: 3122225001073

Question 4:

Consider the problem of making change for n rupees using the smallest number of coins. Assume that each coin's value is an integer.

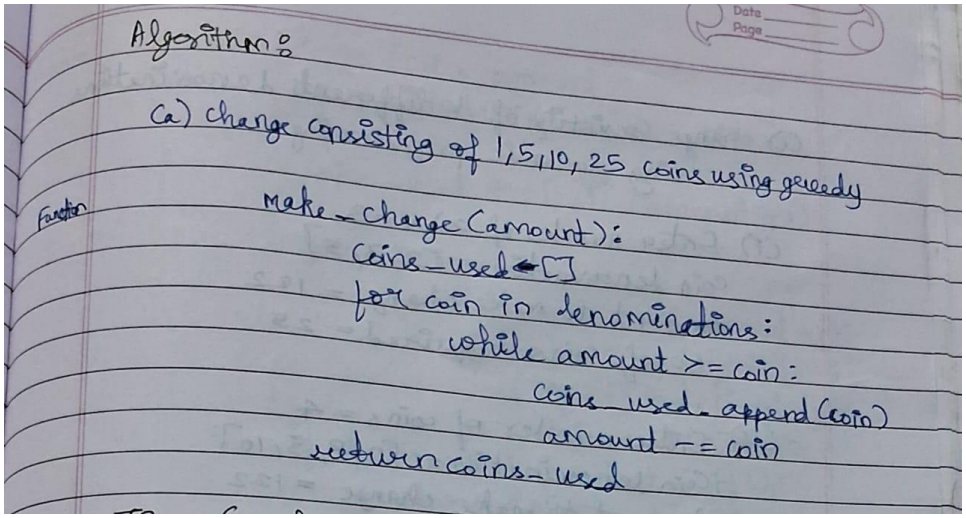
(a) Describe a greedy algorithm to make change consisting of 1, 5, 10, and 25 (hypothetical) coins. Does your algorithm yield an optimal solution? Implement the algorithm in Python.
Example: If you need to make change for 126, then you need to pick five 25 coins and one 1 coin.

(b) Modify the greedy algorithm such that the coin denominations are powers of c, where c > 1. That is, the denominations of coins available are c0, c1, . . . , ck. Does this algorithm give an optimal solution? Implement the algorithm in Python.

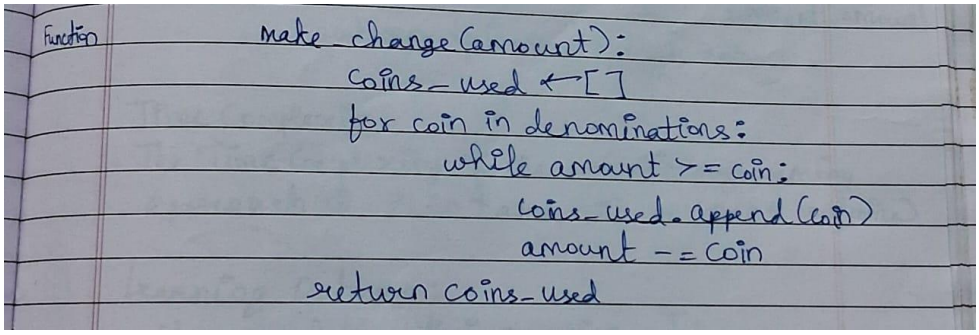
(c) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include 1 so that there is a solution for every value of n (not necessarily optimal).

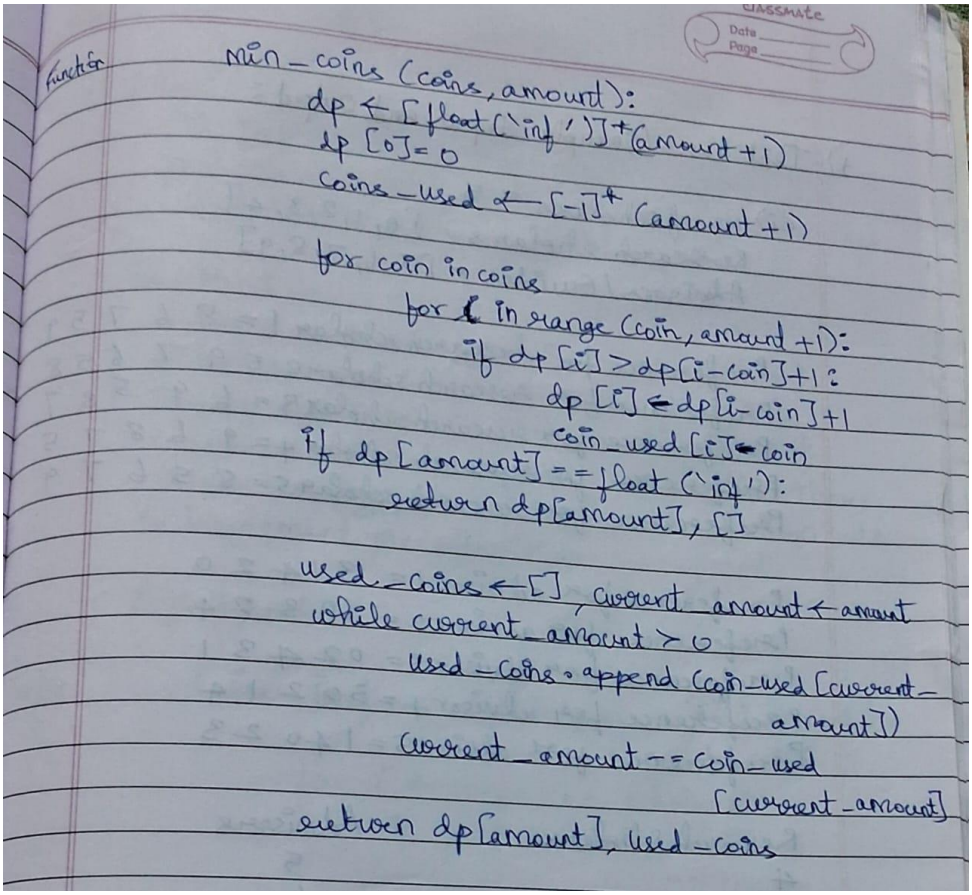
(d) Give an algorithm (not necessarily greedy) that makes change for any set of k different coin denominations using the smallest number of coins, assuming that one of the coins is a 1 coin. Implement the code in Python. What is the time complexity of the algorithm as a function of n alone or as a combination of n and k?

Algorithms:



Name: Mugilkrishna D U
Reg. No.: 3122225001073





CODE:

Code for change consisting of 1, 5, 10 and 25 coins using greedy algorithm:

```
denominations = [25, 10, 5, 1]

def make_change(amount):
    coins_used = []
    for coin in denominations:
        while amount >= coin:
            coins_used.append(coin)
            amount -= coin
    return coins_used
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
amount = int(input("Enter the amount to make change: "))

print("Coins used:", make_change(amount))
print("Number of coins used: ", len(make_change(amount)))
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.1.py"
Enter the amount to make change: 127
Coins used: [25, 25, 25, 25, 25, 1, 1]
Number of coins used:  7
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.1.py"
Enter the amount to make change: 128
Coins used: [25, 25, 25, 25, 25, 1, 1, 1]
Number of coins used:  8
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.1.py"
Enter the amount to make change: 131
Coins used: [25, 25, 25, 25, 25, 5, 1]
Number of coins used:  7
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity of the Greedy Algorithm with change 1, 5, 10 and 25 is O(n)

Code for change consisting of powers of c using greedy algorithm:

```
c = int(input("Enter the base value of c: "))
amount = int(input("Enter the amount to make change: "))
denominations = []
for i in range(100):
    if c ** i > amount:
        break
    denominations.append(c ** i)

denominations.sort(reverse=True)

def make_change(amount):
    coins_used = []
    for coin in denominations:
        while amount >= coin:
            coins_used.append(coin)
            amount -= coin
    return coins_used
```

```
coins_used = make_change(amount)

print("Coins used:", coins_used)
print("Number of coins used:", len(coins_used))
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.2.py"
Enter the base value of c: 5
Enter the amount to make change: 190
Coins used: [125, 25, 25, 5, 5, 5]
Number of coins used: 6
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.2.py"
Enter the base value of c: 6
Enter the amount to make change: 192
Coins used: [36, 36, 36, 36, 36, 6, 6]
Number of coins used: 7
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.2.py"
Enter the base value of c: 3
Enter the amount to make change: 109
Coins used: [81, 27, 1]
Number of coins used: 3
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity of the Greedy Algortithm with change of powers of c is O(n)

Code for change consisting of k different denominations using dynamic programming approach:

```
def min_coins(coins, amount):
    dp = [float('inf')] * (amount + 1)
    dp[0] = 0
    coin_used = [-1] * (amount + 1)

    for coin in coins:
        for i in range(coin, amount + 1):
            if dp[i] > dp[i - coin] + 1:
                dp[i] = dp[i - coin] + 1
                coin_used[i] = coin

    if dp[amount] == float('inf'):
        return dp[amount], []

    used_coins = []
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
    current_amount = amount
    while current_amount > 0:
        used_coins.append(coin_used[current_amount])
        current_amount -= coin_used[current_amount]

    return dp[amount], used_coins

coins = [1]

denominations_length = int(input("Enter the number of coins (1 is already
included): "))

for i in range(denominations_length):
    denomination = int(input("Enter the coin denomination: "))
    coins.append(denomination)

print("\n")
amount = int(input("Enter the amount to make change: "))
min_num_coins, coins_used = min_coins(coins, amount)
print("Minimum number of coins required:", min_num_coins)
print("List of coins used:", coins_used)
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.3.py"
Enter the number of coins (1 is already included): 3
Enter the coin denomination: 2
Enter the coin denomination: 3
Enter the coin denomination: 5


Enter the amount to make change: 122
Minimum number of coins required: 25
List of coins used: [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 2]
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.3.py"
Enter the number of coins (1 is already included): 4
Enter the coin denomination: 2
Enter the coin denomination: 3
Enter the coin denomination: 5
Enter the coin denomination: 10


Enter the amount to make change: 122
Minimum number of coins required: 13
List of coins used: [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 2]
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppDat
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
a/Local/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/
SSN/SEM4/DESIGN AND ANALYSIS OF ALGORITHMS/LAB/9.4.3.py"
Enter the number of coins (1 is already included): 2
Enter the coin denomination: 5
Enter the coin denomination: 10


Enter the amount to make change: 177
Minimum number of coins required: 20
List of coins used: [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 5, 1, 1]
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

The Time Complexity of the Dynamic Programming approach is O(n * number of coins)

Learning Outcome:

Upon completing this exercise, I have understood the applications of greedy algorithms and it's various uses for solving problems in an effective manner. I have now learnt to implement Dijkstra's algorithm, Prim's algorithm and Kruskal's algorithm. I have also understood how to solve the job scheduling problem and how to solve the knapsack problem using two different methods. I have also learnt how to solve the coin denominations problem using different denominations.