**Exercise - 11**

**Approximation Algorithm**

**Aim:** To write the Python code to solve Knapsack and Travel Salesman Problem using Approximation algorithm.

**Algorithm:**

**Knapsack Problem:**

1. Sort the items in descending order based on their value-to-weight ratio. Calculate the value-to-weight ratio for each item by dividing its value by its weight.

2. Initialize an empty knapsack and set the total value and weight to zero.

3. Iterate through the sorted list of items and add items to the knapsack if adding the item does not exceed the capacity of the knapsack.

4. If adding an item exceeds the capacity of the knapsack, calculate the remaining capacity, and add a fraction of the item that can fit into the knapsack. Update the total value and weight accordingly.

5. Continue this process until all items have been considered.

6. Return the knapsack with the maximum value obtained.

**Travel Salesman Problem:**

1. Start at an arbitrary city (let's call it the starting city).

2. Mark this city as visited and set the current city as the starting city.

3. While there are unvisited cities:

4. Find the nearest unvisited city from the current city.

5. Move to the nearest city and mark it as visited.

6. Update the current city to the nearest city.

7. Once all cities have been visited, return to the starting city to complete the cycle.

**Source Code:**

**Knapsack Problem:**

def knapsack\_approximation(values, weights, capacity):

    n = len(values)

    items = list(range(n))

    ratios = [(values[i] / weights[i], i) for i in range(n)]

    ratios.sort(reverse=True)

    total\_value = 0

    total\_weight = 0

    selected\_items = []

    for ratio, i in ratios:

        if total\_weight + weights[i] <= capacity:

            total\_value += values[i]

            total\_weight += weights[i]

            selected\_items.append(i)

    return total\_value, selected\_items

# User input

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

values = []

weights = []

for i in range(n):

    value = int(input("Enter the value of item {}: ".format(i+1)))

    weight = int(input("Enter the weight of item {}: ".format(i+1)))

    values.append(value)

    weights.append(weight)

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

total\_value, selected\_items = knapsack\_approximation(values, weights, capacity)

print("Selected items:", selected\_items)

print("Total value:", total\_value)

**Travel Salesman Problem:**

import math

def tsp\_nearest\_neighbor(nodes):

    visited = set()

    current\_node = nodes[0]

    visited.add(current\_node)

    path = [current\_node]

    while len(visited) < len(nodes):

        next\_node = None

        min\_distance = float('inf')

        for node in nodes:

            if node not in visited:

                distance = math.dist(current\_node, node)

                if distance < min\_distance:

                    min\_distance = distance

                    next\_node = node

        current\_node = next\_node

        visited.add(current\_node)

        path.append(current\_node)

    return path

# Get the number of nodes from the user

num\_nodes = int(input("Enter the number of nodes: "))

# Prompt the user for the coordinates of each node

nodes = []

for i in range(num\_nodes):

    x = float(input(f"Enter the x-coordinate for node {i+1}: "))

    y = float(input(f"Enter the y-coordinate for node {i+1}: "))

    nodes.append((x, y))

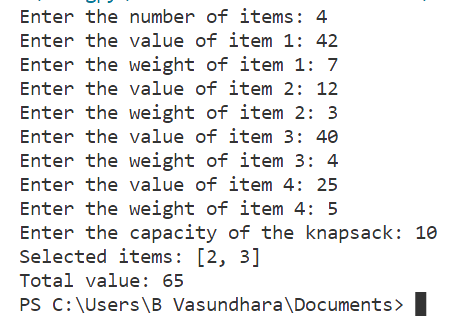
# Compute the TSP path using the nearest neighbor algorithm

path = tsp\_nearest\_neighbor(nodes)

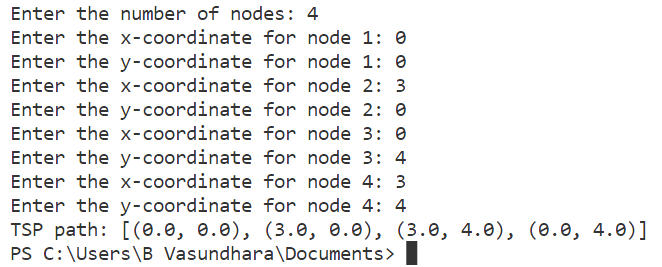
print("TSP path:", path)

**Sample input and output:**

**Knapsack Problem:**



**Travel Salesman Problem:**



**Result:**

Thus, an Approximation algorithm in Python to solve Knapsack Problem and Travel Salesman Problem has been successfully implemented and the output is verified.