	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of Artificial Intelligence and Data Science**

Student Name: Mohammad Faiz Nishat Parvej Saiyad		
Class: TY	Division: A	Roll No:371034
Semester: V	Academic Year:2022-23	
Subject Name & Code: Design and Analysis of Algorithms		
Title of Assignment: **Solve the following instance of the knapsack problem given the knapsack capacity in w=20 using greedy methods. Total no of item is 5.** **Item Weight Profit** **X1 3 10** **X1 5 20** **X1 5 21** **X1 8 30** **X1 4 16**		
Date of Performance:	Date of Submission:	

Aim:

Solve the following instance of the knapsack problem given the knapsack capacity in w=20 using greedy methods. Total no of item is 5.

Item	Weight	Profit
X1	3	10
X1	5	20
X1	5	21
X1	8	30
X1	4	16

Problem Statement:

To solve Knapsack problem using Greedy Algorithmic Approach

Software Requirements:

Text Editor: VSCode, Neovim, etc

Environment: Python 3.10

Terminal Emulator

Background Information:

Greedy Algorithm:

Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. So the problems where choosing locally optimal also leads to global solution are the best fit for Greedy.

For example consider the Fractional Knapsack Problem. The local optimal strategy is to choose the item that has maximum value vs weight ratio. This strategy also leads to a globally optimal solution because we are allowed to take fractions of an item.

Fractional Knapsack Problem:

Given the weights and values of N items, in the form of {value, weight} put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In Fractional Knapsack, we can break items for maximizing the total value of the knapsack

Follow the given steps to solve the problem using the above approach:

- Calculate the ratio(value/weight) for each item.
- Sort all the items in decreasing order of the ratio.
- Initialize res =0, curr_cap = given_cap.
- Do the following for every item “i” in the sorted order:
- If the weight of the current item is less than or equal to the remaining capacity then add the value of that item into the result
- Else add the current item as much as we can and break out of the loop.
- Return res.

Code:

~/D/D/S/S/2/A/0/Knapsack-without-Args.py

```python
from typing import List

class Node:
    def __init__(self, p, w) -> None:
        self.p = p
        self.w = w
        self.pbw = p / w
        self.x = 0

    def profit(self):
        return self.p * self.x

    def __repr__(self) -> str:
        return f"Profit: {self.p}; Weight: {self.w}; Profit by Weight: {self.pbw}; Portion: {self.x}"


def sorter(a):
    return sorted(a, key=lambda x: x.pbw)


def parts(M: int, w: int, wo: int):
    return (M - w) / wo


def print_result(arr: List):
    from rich.table import Table
    from rich import print

    tb = Table()
    tb.add_column("Weight (W)")
    tb.add_column("Profit (P)")
    tb.add_column("Profit By Weight (P/W)")
    tb.add_column("Portion (X)")

    profit = 0
    for node in arr:
        tb.add_row(str(node.w), str(node.p), str(node.pbw), str(node.x))
        profit += node.profit()

    print(tb)

    print(f"Total profit: [blue]{profit}")

```

```python
~/D/D/S/S/2/A/0/Knapsack-without-Args.py
def print_normal(arr: List):

    profit = 0
    for node in arr:
        print(node)
        profit += node.profit()

    print(f"Total profit: {profit}")


def main():

    n = int(input("Enter n: "))
    M = int(input("Enter M: "))

    done = False

    arr = []

    for i in range(0, n):     ■ "i" is not accessed
        inp = input().split()

        p, w = (int(i) for i in inp)
        arr.append(Node(p, w))

    arr = sorter(arr)
    arr.reverse()

    agg = 0
    for node in arr:
        if done == True:
            break

        if node.w >= M or node.w + agg >= M:
            node.x = parts(M, agg, node.w)
            break

        agg += node.w
        node.x = 1

    print_result(arr)
    # print_normal(arr)

88
if __name__ == "__main__":
    main()
```

Output:

```
2. Design and Analysis of Algorithm/Assignments/03. Knapsack Problem via  v3.10.7
❯ python args.py
```

Weight (W)	Profit (P)	Profit By Weight (P/W)	Portion (X)
5	21	4.2	1
4	16	4.0	1
5	20	4.0	1
8	30	3.75	0.75
3	10	3.3333333333333335	0

```
Total profit: 79.5

2. Design and Analysis of Algorithm/Assignments/03. Knapsack Problem via  v3.10.7
❯
```

Conclusion:

Implemented Fractional Knapsack problem using the Greedy Method.