



# Tecnológico de Monterrey

Tecnológico de Monterrey - Campus Monterrey  
School of Engineering and Sciences  
Engineering in Computational Technologies  
Analysis and Design of Advanced Algorithms

## Homework 4: Knapsack Problem using Greedy Algorithm

Group: 607  
Team #3  
Dr. Katie Brodhead

Santiago Quintana Moreno A01571222  
Miguel Ángel Álvarez Hermida a01722925

FileEditSelectionViewGoRunTerminalHelp

2.Advanced Algorithms

GoldMine\_Homework3.pyKnapsackWGreedy.py

GreedyAlgorithms > KnapsackWGreedy.py > ...  
1 # Analysis and Design of Advanced Algorithms  
2 # Group #607  
3 # Team 3  
4 # Dr. Katie Brodhead  
5  
6 # Santiago Quintana Moreno A01571222  
7 # Miguel Ángel Álvarez Hermida A01722925  
8  
9 # ----- KNAPSACK PROBLEM - GREEDY ALGORITHMS -----  
10  
11 # Total Complexity:  $O(n \log n)$  time,  $O(n)$  space - dominated by sorting operations  
12  
13  
14 def knapsack\_greedy(weights, values, capacity):  
15 # Solves the fractional knapsack problem using a greedy algorithm.  
16 #  
17 # Time Complexity:  $O(n \log n)$  - due to sorting  
18 # Space Complexity:  $O(n)$  - for storing items with ratios  
19  
20  
21 n = len(weights)  
22  
23 # Create list of items with value-to-weight ratio  
24 items = []  
25 for i in range(n):  
26 if weights[i] > 0: # Avoid division by zero  
27 ratio = values[i] / weights[i]  
28 items.append((ratio, weights[i], values[i], i))  
29  
30 # Sort items by value-to-weight ratio in descending order  
31 items.sort(reverse=True, key=lambda x: x[0])  
32  
33 max\_value = 0  
34 selected\_items = []  
35 remaining\_capacity = capacity  
36  
37 # Greedily select items  
38 for ratio, weight, value, index in items:  
39 if weight <= remaining\_capacity:  
40 # Take the entire item  
41 max\_value += value  
42 remaining\_capacity -= weight

powershell  
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> &  
C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d  
:/1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms/Greedy  
Algorithms/KnapsackWGreedy.py"  
Items:  
Item 0: weight=10, value=60, ratio=6.00  
Item 1: weight=40, value=40, ratio=1.00  
Item 2: weight=20, value=100, ratio=5.00  
Item 3: weight=30, value=120, ratio=4.00  
  
Knapsack capacity: 50  
-----  
Fractional Knapsack Solution:  
Maximum value: 240.00  
Selected items (index, fraction):  
Item 0: 1.00 (value: 60.00)  
Item 2: 1.00 (value: 100.00)  
Item 3: 0.67 (value: 80.00)  
  
0-1 Knapsack Greedy Solution:  
Maximum value: 160  
Selected items (indices): [0, 2]  
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>

SantiQ (2 minutes ago)Ln 1, Col 46Spaces: 4UTF-8CRLF{}PythonPython 3.12Go Live

FileEditSelectionViewGoRunTerminalHelp

2.Advanced Algorithms

GoldMine\_Homework3.pyKnapsackWGreedy.py

GreedyAlgorithms > KnapsackWGreedy.py > ...

```
14 def knapsack_greedy(weights, values, capacity):
42     remaining_capacity = weight
43     selected_items.append((index, 1.0)) # (item_index, fraction_taken)
44     elif remaining_capacity > 0:
45         # Take fraction of the item (for fractional knapsack)
46         fraction = remaining_capacity / weight
47         max_value += value * fraction
48         selected_items.append((index, fraction))
49         remaining_capacity = 0
50         break
51
52     return max_value, selected_items
53
54 def knapsack_01_greedy(weights, values, capacity):
55     # Solves the 0-1 knapsack problem using a greedy approximation.
56     # Note: This doesn't guarantee optimal solution for 0-1 knapsack.
57     #
58     # Time Complexity: O(n log n)
59     # Space Complexity: O(n)
60     n = len(weights)
61
62     # Create list of items with value-to-weight ratio
63     items = []
64     for i in range(n):
65         if weights[i] > 0:
66             ratio = values[i] / weights[i]
67             items.append((ratio, weights[i], values[i], i))
68
69     # Sort by ratio in descending order
70     items.sort(reverse=True, key=lambda x: x[0])
71
72     max_value = 0
73     selected_items = []
74     remaining_capacity = capacity
75
76     # Select items that fit completely
77     for ratio, weight, value, index in items:
78         if weight <= remaining_capacity:
79             max_value += value
80             remaining_capacity -= weight
81             selected_items.append(index)
82
```

```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> &
C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d
:/1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms/Greedy
Algorithms/KnapsackWGreedy.py"
Items:
Item 0: weight=10, value=60, ratio=6.00
Item 1: weight=40, value=40, ratio=1.00
Item 2: weight=20, value=100, ratio=5.00
Item 3: weight=30, value=120, ratio=4.00

Knapsack capacity: 50
-----
Fractional Knapsack Solution:
Maximum value: 240.00
Selected items (index, fraction):
Item 0: 1.00 (value: 60.00)
Item 2: 1.00 (value: 100.00)
Item 3: 0.67 (value: 80.00)

0-1 Knapsack Greedy Solution:
Maximum value: 160
Selected items (indices): [0, 2]
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```

SantiQ (2 minutes ago)

Ln 1, Col 46

Spaces: 4

UTF-8

CRLF

{}

Python

Python 3.12

Go Live

FileEditSelectionViewGoRunTerminalHelp←→2.Advanced Algorithms

GoldMine\_Homework3.pyKnapsackWGreedy.py

GreedyAlgorithms > KnapsackWGreedy.py > ...

```
54 def knapsack_01_greedy(weights, values, capacity):
81     selected_items.append(index)
82
83     return max_value, selected_items
84
85 def print_solution(weights, values, capacity, solution_type="fractional"):
86     # Helper function to print the knapsack solution.
87
88     if solution_type == "fractional":
89         max_value, selected_items = knapsack_greedy(weights, values, capacity)
90         print(f"Fractional Knapsack Solution:")
91         print(f"Maximum value: {max_value:.2f}")
92         print("Selected items (index, fraction):")
93         for item_index, fraction in selected_items:
94             print(f"    Item {item_index}: {fraction:.2f} (value: {values[item_index] * fraction:.2f})")
95     else:
96         max_value, selected_items = knapsack_01_greedy(weights, values, capacity)
97         print(f"0-1 Knapsack Solution:")
98         print(f"Maximum value: {max_value}")
99         print("Selected items (indices):", selected_items)
100
101 # Example usage
102 if __name__ == "__main__":
103     # Example data
104     weights = [10, 40, 20, 30]
105     values = [60, 40, 100, 120]
106     capacity = 50
107
108     print("Items:")
109     for i in range(len(weights)):
110         ratio = values[i] / weights[i]
111         print(f"Item {i}: weight={weights[i]}, value={values[i]}, ratio={ratio:.2f}")
112
113     print(f"\nKnapsack capacity: {capacity}")
114     print("-" * 40)
115
116     # Solve fractional knapsack
117     print_solution(weights, values, capacity, "fractional")
118     print()
119
120     # Solve 0-1 knapsack (greedy approximation)
121     print_solution(weights, values, capacity, "01")
```

powershell

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3.12.exe "d:/1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms/Greedy Algorithms/KnapsackWGreedy.py"

Items:

Item 0: weight=10, value=60, ratio=6.00

Item 1: weight=40, value=40, ratio=1.00

Item 2: weight=20, value=100, ratio=5.00

Item 3: weight=30, value=120, ratio=4.00

Knapsack capacity: 50

-----

Fractional Knapsack Solution:

Maximum value: 240.00

Selected items (index, fraction):

Item 0: 1.00 (value: 60.00)

Item 2: 1.00 (value: 100.00)

Item 3: 0.67 (value: 80.00)

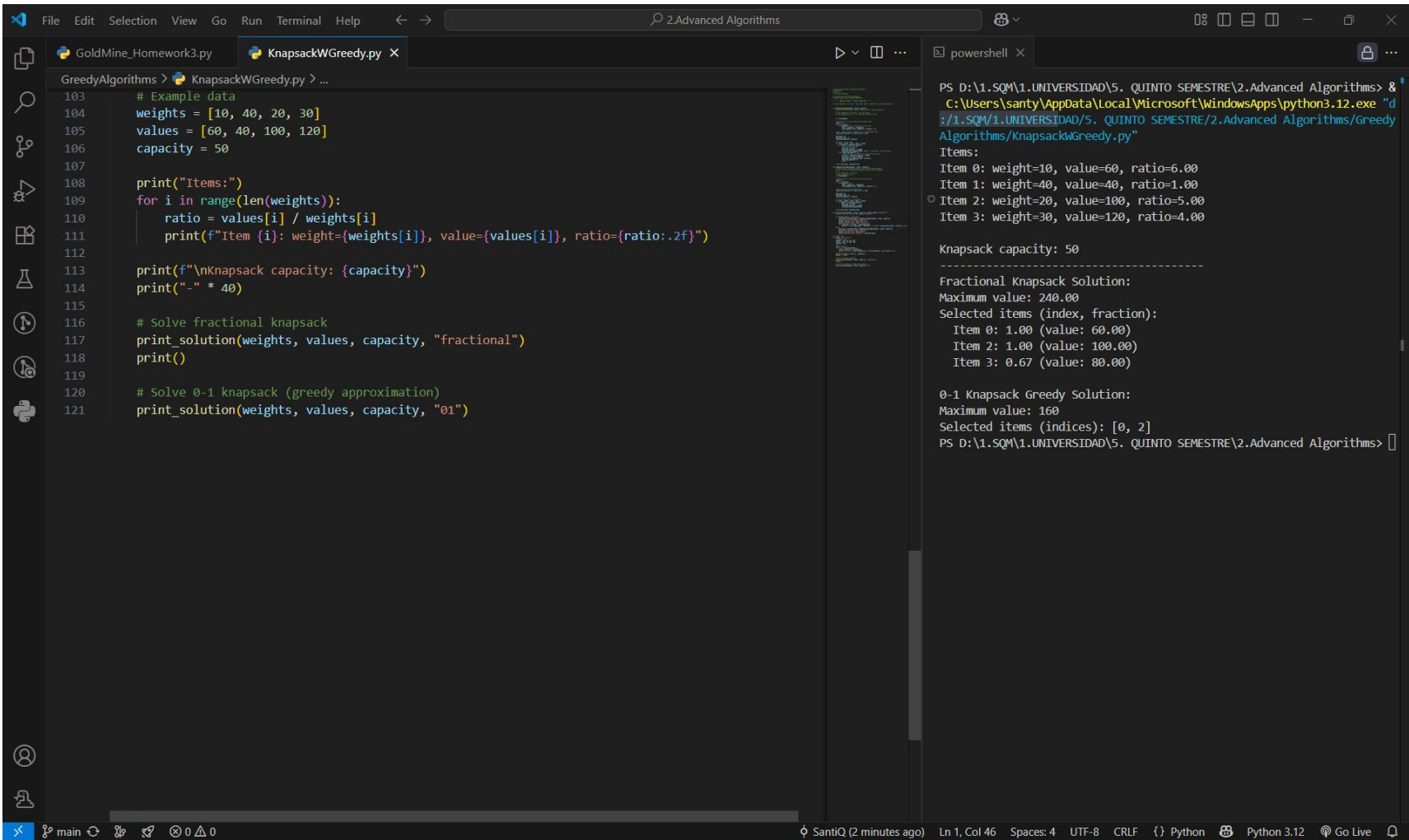
0-1 Knapsack Greedy Solution:

Maximum value: 160

Selected items (indices): [0, 2]

PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>

SantiQ (2 minutes ago)Ln 1, Col 46Spaces: 4UTF-8CRLF{}PythonPython 3.12Go Live



[https://colab.research.google.com/drive/1rWqSihblBB6cZ3ZsHH01WmYgTsd\\_mhJV?usp=sharing](https://colab.research.google.com/drive/1rWqSihblBB6cZ3ZsHH01WmYgTsd_mhJV?usp=sharing)

# REFERENCES

GeeksforGeeks. (2025, July 23). *0/1 Knapsack problem*. GeeksforGeeks.

<https://www.geeksforgeeks.org/dsa/0-1-knapsack-problem-dp-10/>

W3Schools.com. (n.d.). [https://www.w3schools.com/dsa/dsa\\_ref\\_greedy.php](https://www.w3schools.com/dsa/dsa_ref_greedy.php)

GeeksforGeeks. (2025, July 25). *Greedy algorithms*. GeeksforGeeks. <https://www.geeksforgeeks.org/dsa/greedy-algorithms/>

*Greedy Algorithms* | Brilliant Math & Science Wiki. (n.d.). <https://brilliant.org/wiki/greedy-algorithm/>