




Shopping Optimization

Using Kanpsack Algorithm

Submitted by:-

Ayushi Agarwal (RA2211003011089)


Jhanvi Gupta (RA2211003011090)





ABSTRACT

Grocery shoppers encounter difficulties in efficiently utilizing their shopping cart space due to varying item sizes and values. The Knapsack problem emerges, where users aim to optimize their purchases within limited cart capacity while maximizing value. Developing a solution that suggests the most valuable items for the available space is imperative to enhance shopping experiences.



PROBLEM STATEMENT


Grocery shoppers encounter difficulties in efficiently utilizing their shopping cart space due to varying item sizes and values. The Knapsack problem emerges, where users aim to optimize their purchases within limited cart capacity while maximizing value. Developing a solution that suggests the most valuable items for the available space is imperative to enhance shopping experiences.






ALGORITHM USED

Knapsack algorithm is used which is based on the dynamic programming approach. It involves breaking the problem into subproblems and solving each subproblem only once, storing the solution to the subproblem in table for later use. The basic idea is to consider each item one at a time and determine whether it should be included in the knapsack or not.





WHY ALGORITHM USED?

- In the context of the shopping app, the Knapsack algorithm allows users to optimize their shopping lists by selecting items based on their weight and value while staying within the weight limit of their shopping cart.
 - It provides efficient way to solve problem.
 - Offers balance between computational complexity and effectiveness in finding the optimal solution.
- 

CODE

```
File Edit Selection View Go Run Terminal Help
Welcome 2.py X
C:\Users> Shubham > Downloads > 2.py > ...
1 import tkinter as tk
2 from tkinter import ttk
3
4 class Item:
5     def __init__(self, name, weight, value):
6         self.name = name
7         self.weight = weight
8         self.value = value
9
10 items = []
11 Item("Apple", 0.5, 1),
12 Item("Banana", 0.3, 0.5),
13 Item("Orange", 0.4, 0.7),
14 Item("Grapes", 0.6, 1.2),
15 Item("Milk", 1.0, 1.5),
16 Item("Bread", 0.8, 1.0),
17 Item("Cheese", 0.7, 1.3),
18 Item("Chicken", 1.2, 2.0),
19 Item("Eggs", 0.2, 0.3),
20 Item("Tomato", 0.3, 0.4), # Additional items
21 Item("Potato", 0.4, 0.6),
22 Item("Carrot", 0.3, 0.5),
23 Item("Spinach", 0.2, 0.3),
24 Item("Onion", 0.3, 0.4),
25 Item("Cucumber", 0.4, 0.5),
26 Item("Watermelon", 1.5, 2.0),
27 Item("Pineapple", 0.8, 1.2),
28 Item("Strawberry", 0.2, 0.4),
29
30
31 weight_limit = 3.0 # Weight limit for carrying capacity
```

```
File Edit Selection View Go Run Terminal Help
Welcome 2.py X
C:\Users> Shubham > Downloads > 2.py > ...
33 class ShoppingApp(tk.Tk):
34     def __init__(self):
35         super().__init__()
36         self.title("Supermarket Shopping Optimization")
37         self.geometry("800x600") # Set the window size
38
39         self.selected_items = []
40
41         # Main frame
42         self.main_frame = tk.Frame(self)
43         self.main_frame.pack(fill=tk.BOTH, expand=True) # Fill the entire window
44
45         # Canvas for displaying items
46         self.canvas = tk.Canvas(self.main_frame, bg="white")
47         self.canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
48
49         # Frame for controls
50         self.control_frame = tk.Frame(self.main_frame)
51         self.control_frame.pack(side=tk.RIGHT, fill=tk.Y)
52
53         # Label for available items
54         self.items_label = tk.Label(self.control_frame, text="Available Items:", font=("Helvetica", 12))
55         self.items_label.pack(padx=5, pady=5, anchor="w")
56
57         # Combobox for selecting items
58         self.items_dropdown = ttk.Combobox(self.control_frame, values=[item.name for item in items], state="readonly")
59         self.items_dropdown.pack(padx=5, pady=5)
60
61         # Button to add an item
62         add_button = ttk.Button(self.control_frame, text="Add", command=self.add_item)
63         add_button.pack(padx=5, pady=5)
64
65         # Label for selected items
66         self.selected_items_label = tk.Label(self.control_frame, text="Selected Items:", font=("Helvetica", 12))
67         self.selected_items_label.pack(padx=5, pady=5, anchor="w")
```


CODE

```
File Edit Selection View Go Run Terminal Help Python Debugger: Curre...
Welcome 2.py x

C:\Users\Shubham\Downloads> 2.py > ...
33 class ShoppingApp(tk.Tk):
34     def __init__(self):
35         # Initialize items
36
37         # Label for total selected items
38         self.update_selected_items()
39         self.selected_items_label.config(text="Selected Items:\n" + selected_items_text)
40         self.selected_items_label.pack(padx=5, pady=5, anchor="w")
41
42         # Label for remaining weight capacity
43         self.remaining_weight_label = tk.Label(self.control_frame, text="", font=("Helvetica", 12))
44         self.remaining_weight_label.pack(padx=5, pady=5, anchor="w")
45
46         # Button to remove an item
47         remove_button = ttk.Button(self.control_frame, text="Remove", command=self.remove_item)
48         remove_button.pack(padx=5, pady=5)
49
50         # Button to optimize the shopping list
51         optimize_button = ttk.Button(self.control_frame, text="Optimize", command=self.optimize_shopping_list)
52         optimize_button.pack(padx=5, pady=5)
53
54         # Draw knapsack table
55         self.draw_knapsack_table()
56
57     def add_item(self):
58         item_name = self.items_dropdown.get()
59         item = next((item for item in items if item.name == item_name), None)
60         if item:
61             self.selected_items.append(item)
62             self.update_selected_items()
63
64     def remove_item(self):
65         if self.selected_items:
66             self.selected_items.pop()
67             self.update_selected_items()
```

```
File Edit Selection View Go Run Terminal Help Python Debugger: Curre...
Welcome 2.py x

C:\Users\Shubham\Downloads> 2.py > ...
33 class ShoppingApp(tk.Tk):
34
35     def update_selected_items(self):
36         selected_items_text = "\n".join([item.name for item in self.selected_items])
37         total_weight = sum([item.weight for item in self.selected_items])
38         self.selected_items_label.config(text="Selected Items:\n" + selected_items_text)
39         self.total_weight_label.config(text=f"Total Weight: {total_weight} kg")
40         remaining_weight = max(weight_limit - total_weight, 0)
41         self.remaining_weight_label.config(text=f"Remaining Weight Capacity: {remaining_weight} kg")
42
43     def optimize_shopping_list(self):
44         total_weight = sum([item.weight for item in self.selected_items])
45         total_value = sum([item.value for item in self.selected_items])
46         if total_weight > weight_limit:
47             self.selected_items_label.config(text="Cannot optimize: Total weight exceeds limit")
48         else:
49             self.selected_items_label.config(text=f"Total Weight: {total_weight} kg\nTotal Value: {total_value}")
50
51     def draw_knapsack_table(self):
52         self.canvas.create_text(20, 20, text="Item", anchor="w")
53         self.canvas.create_text(120, 20, text="Weight (kg)", anchor="w")
54         self.canvas.create_text(220, 20, text="Value", anchor="w")
55
56         y_offset = 40
57         for item in items:
58             self.canvas.create_text(20, y_offset, text=item.name, anchor="w")
59             self.canvas.create_text(120, y_offset, text=item.weight, anchor="w")
60             self.canvas.create_text(220, y_offset, text=item.value, anchor="w")
61             y_offset += 20
62
63 if __name__ == "__main__":
64     app = ShoppingApp()
65     app.mainloop()
```

OUTPUT

Before Optimization

Supermarket Shopping Optimization

Item	Weight (kg)	Value
Apple	0.5	1
Banana	0.3	0.5
Orange	0.4	0.7
Grapes	0.6	1.2
Milk	1.0	1.5
Bread	0.8	1.0
Cheese	0.7	1.3
Chicken	1.2	2.0
Eggs	0.2	0.3
Tomato	0.3	0.4
Potato	0.4	0.6
Carrot	0.3	0.5
Spinach	0.2	0.3
Onion	0.3	0.4
Cucumber	0.4	0.5
Watermelon	1.5	2.0
Pineapple	0.8	1.2
Strawberry	0.2	0.4

Available Items:

Selected Items:

Apple
Grapes
Bread
Cheese

Total Weight: 2.6 kg

Remaining Weight Capacity: 0.3999999999999999 kg

After Optimization

Supermarket Shopping Optimization

Item	Weight (kg)	Value
Apple	0.5	1
Banana	0.3	0.5
Orange	0.4	0.7
Grapes	0.6	1.2
Milk	1.0	1.5
Bread	0.8	1.0
Cheese	0.7	1.3
Chicken	1.2	2.0
Eggs	0.2	0.3
Tomato	0.3	0.4
Potato	0.4	0.6
Carrot	0.3	0.5
Spinach	0.2	0.3
Onion	0.3	0.4
Cucumber	0.4	0.5
Watermelon	1.5	2.0
Pineapple	0.8	1.2
Strawberry	0.2	0.4

Available Items:

Total Weight: 2.6 kg
Total Value: 4.5

Total Weight: 2.6 kg

Remaining Weight Capacity: 0.3999999999999999 kg

OUTPUT

Exceed Stack size

Supermarket Shopping Optimization

Item	Weight (kg)	Value
Apple	0.5	1
Banana	0.3	0.5
Orange	0.4	0.7
Grapes	0.6	1.2
Milk	1.0	1.5
Bread	0.8	1.0
Cheese	0.7	1.3
Chicken	1.2	2.0
Eggs	0.2	0.3
Tomato	0.3	0.4
Potato	0.4	0.6
Carrot	0.3	0.5
Spinach	0.2	0.3
Onion	0.3	0.4
Cucumber	0.4	0.5
Watermelon	1.5	2.0
Pineapple	0.8	1.2
Strawberry	0.2	0.4

Available Items:

Add

Selected Items:

Apple
Grapes
Bread
Cheese
Eggs
Milk
Grapes

Total Weight: 4.4 kg

Remaining Weight Capacity: 0 kg

Remove

Optimize

Optimization

Supermarket Shopping Optimization

Item	Weight (kg)	Value
Apple	0.5	1
Banana	0.3	0.5
Orange	0.4	0.7
Grapes	0.6	1.2
Milk	1.0	1.5
Bread	0.8	1.0
Cheese	0.7	1.3
Chicken	1.2	2.0
Eggs	0.2	0.3
Tomato	0.3	0.4
Potato	0.4	0.6
Carrot	0.3	0.5
Spinach	0.2	0.3
Onion	0.3	0.4
Cucumber	0.4	0.5
Watermelon	1.5	2.0
Pineapple	0.8	1.2
Strawberry	0.2	0.4

Available Items:

Add

Cannot optimize: Total weight exceeds limit

Total Weight: 4.4 kg

Remaining Weight Capacity: 0 kg

Remove

Optimize



THANK YOU

