

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

```
□ II 🦪 🖠 ↑ 🖰 🔲 Python Debugger: Curre ∨
File Edit Selection View Go Run Terminal Help
  ₩elcome
                2.py X
  C: > Users > Shubham > Downloads > * 2.py > ...
    1 import tkinter as tk
    2 from tkinter import ttk
            def __init__(self, name, weight, value):
                self.name = name
                self.weight = weight
                self.value = value
        items =
            Item("Apple", 0.5, 1),
            Item("Banana", 0.3, 0.5),
            Item("Orange", 0.4, 0.7),
            Item("Grapes", 0.6, 1.2),
            Item("Milk", 1.0, 1.5),
            Item("Bread", 0.8, 1.0),
            Item("Cheese", 0.7, 1.3),
            Item("Chicken", 1.2, 2.0),
            Item("Eggs", 0.2, 0.3),
            Item("Potato", 0.4, 0.6),
            Item("Carrot", 0.3, 0.5),
            Item("Spinach", 0.2, 0.3),
            Item("Onion", 0.3, 0.4),
            Item("Cucumber", 0.4, 0.5),
            Item("Watermelon", 1.5, 2.0),
            Item("Pineapple", 0.8, 1.2),
            Item("Strawberry", 0.2, 0.4),
        weight limit = 3.0 # Weight limit for carrying capacity
```

```
□ II 🦪 † 💲 🗖 Python Debugger: Curre ∨
File Edit Selection View Go Run Terminal Help
  Welcome
              def init (self):
                  super(). init ()
                  self.title("Supermarket Shopping Optimization")
                  self.geometry("800x600") # Set the window size
                  self.selected items = []
                  self.main frame = tk.Frame(self)
                  self.main frame.pack(fill=tk.BOTH, expand=True) # Fill the entire window
                  self.canvas = tk.Canvas(self.main frame, bg="white")
                  self.canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
                  self.control frame = tk.Frame(self.main frame)
                  self.control frame.pack(side=tk.RIGHT, fill=tk.Y)
                  self.items_label = tk.Label(self.control_frame, text="Available Items:", font=("Helvetica", 12))
                  self.items_label.pack(padx=5, pady=5, anchor="w")
                  self.items dropdown = ttk.Combobox(self.control frame, values=[item.name for item in items], state="readonly")
                  self.items dropdown.pack(padx=5, pady=5)
                  add_button = ttk.Button(self.control_frame, text="Add", command=self.add_item)
                  add button.pack(padx=5, pady=5)
                  self.selected items label = tk.Label(self.control frame, text="Selected Items:", font=("Helvetica", 12))
                  self.selected_items_label.pack(padx=5, pady=5, anchor="w")
```

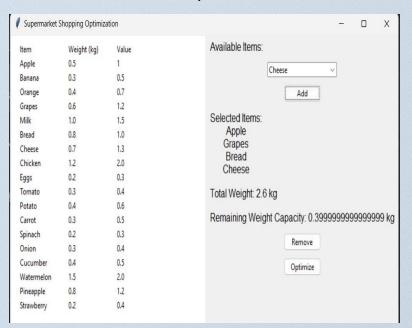
CODE

```
omo 🚁 270m nomo — 270m Ahmo? 💠 🗘 🖰 Pvilion Debuuger Curre 🗸
File Felix Selection "View - Co"Thir Tomana "Title"
                # Label! [voltotel Stimble-0]
                rolfselfrägdalgiseletted items() "'folk control form tout "" fort ("naturation" and
                self.total weight label.pack(padx=5, pady=5, anchor="w")
                self.remaining weight label = tk.Label(self.control frame, text="", font=("Helvetica", 12))
                self.remaining weight label.pack(padx=5, pady=5, anchor="w")
                remove button = ttk.Button(self.control frame, text="Remove", command=self.remove item)
                remove button.pack(padx=5, pady=5)
                optimize button = ttk.Button(self.control frame, text="Optimize", command=self.optimize shopping list)
                optimize button.pack(padx=5, pady=5)
                self.draw knapsack table()
            def add item(self):
                item name = self.items dropdown.get()
                item = next((item for item in items if item.name == item name), None)
                if item:
                    self.selected items.append(item)
                    self.update selected items()
            def remove item(self):
                if self.selected items:
                    self.selected items.pop()
                    self.update selected items()
```

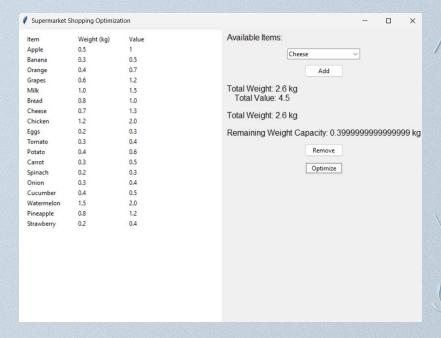
```
□ II 🖓 🙏 🗘 🕽 🔲 Python Debugger: Curre∨
XI File Edit Selection View Go Run Terminal Help
     ₩ Welcome
                     2.py X
                 def update selected items(self):
                     selected items text = "\n".join([item.name for item in self.selected items])
                     total weight = sum([item.weight for item in self.selected items])
                     self.selected items label.config(text="Selected Items:\n" + selected items text)
                     self.total weight label.config(text=f"Total Weight: {total weight} kg")
                     remaining weight = max(weight limit - total weight, 0)
                     self.remaining weight label.config(text=f"Remaining Weight Capacity: {remaining weight} kg")
                 def optimize shopping list(self):
                     total weight = sum([item.weight for item in self.selected items])
                     total value = sum([item.value for item in self.selected items])
                    if total weight > weight limit:
                         self.selected items label.config(text="Cannot optimize: Total weight exceeds limit")
                         self.selected_items_label.config(text=f"Total_Weight: {total_weight} kg\nTotal_Value: {total_value}")
                 def draw knapsack table(self):
                     self.canvas.create text(20, 20, text="Item", anchor="w")
                     self.canvas.create_text(120, 20, text="Weight (kg)", anchor="w")
                     self.canvas.create text(220, 20, text="Value", anchor="w")
                     v offset = 40
                     for item in items:
                         self.canvas.create text(20, y offset, text=item.name, anchor="w")
                         self.canvas.create text(120, y offset, text=item.weight, anchor="w")
                         self.canvas.create text(220, y offset, text=item.value, anchor="w")
                         y offset += 20
             if name == " main ":
                 app.mainloop()
```

OUTPUT

Before Optimization

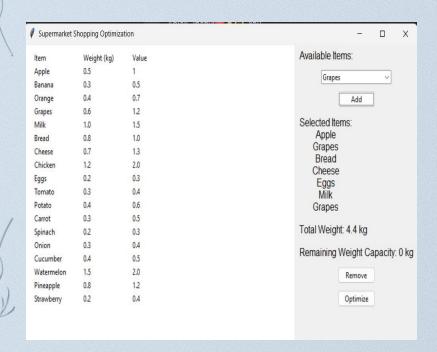


After Optimization



OUTPUT

Exceed Stack size



Optimization

			Aveilable Harrey
ltem	Weight (kg)	Value	Available Items:
Apple	0.5	1	Grapes Add Cannot optimize: Total weight exceeds limit Total Weight: 4.4 kg Remaining Weight Capacity: 0 kg Remove Optimize
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	

THANK YOU



