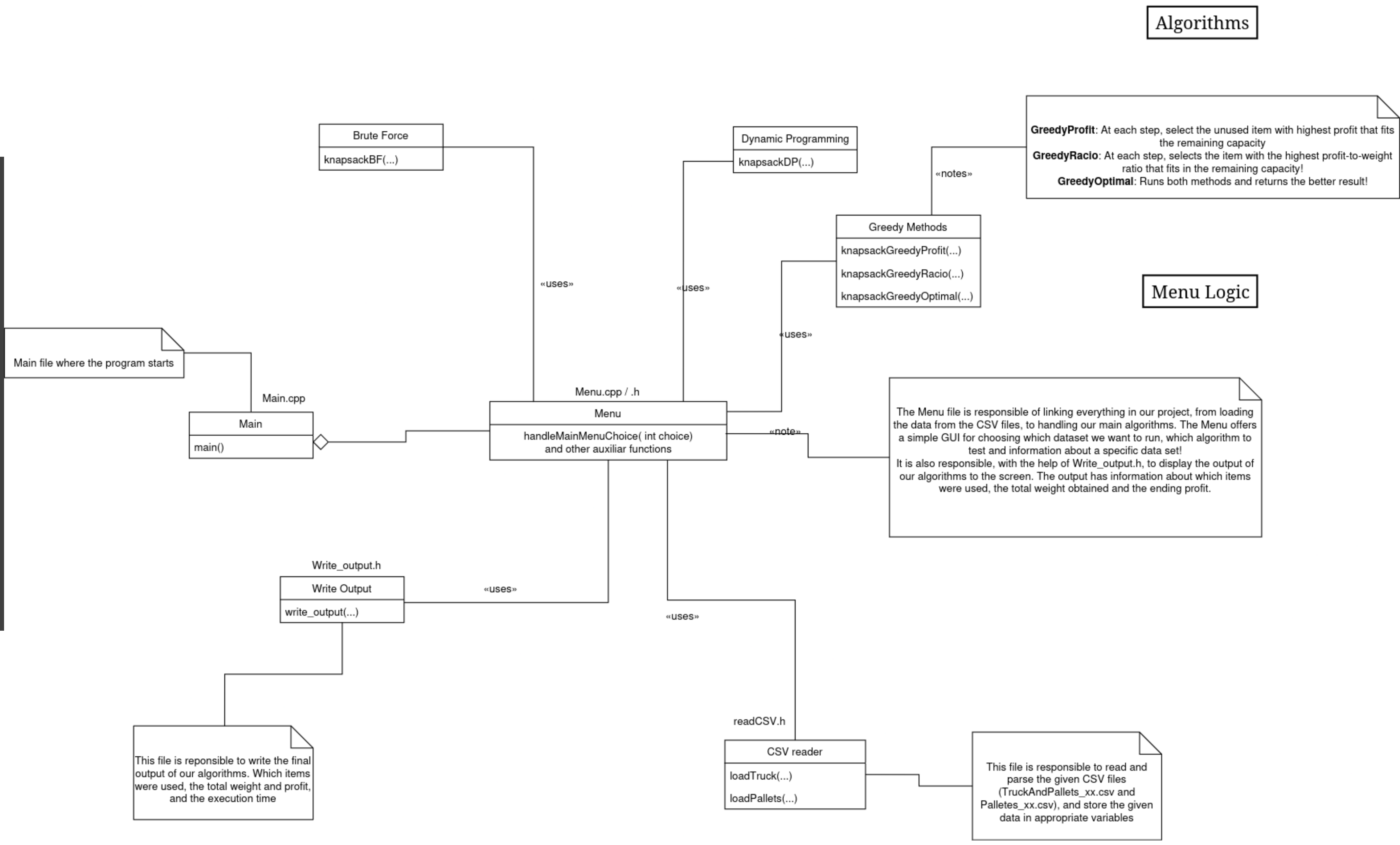
DA Project2 Presentation

<u>Project done by:</u>

- Gustavo Lourenço 202306578
- Tomás Sousa 202303524
- Gonçalo França 202305533

The UML diagram



GreedyProfit: At each step, select the unused item with highest profit that fits the remaining capacity
GreedyRacio: At each step, selects the item with the highest profit-to-weight ratio that fits in the remaining capacity!
GreedyOptimal: Runs both methods and returns the better result!

The Menu file is responsible of linking everything in our project, from loading the data from the CSV files, to handling our main algorithms. The Menu offers a simple GUI for choosing which dataset we want to run, which algorithm to test and information about a specific data set!
It is also responsible, with the help of Write_output.h, to display the output of our algorithms to the screen. The output has information about which items were used, the total weight obtained and the ending profit.

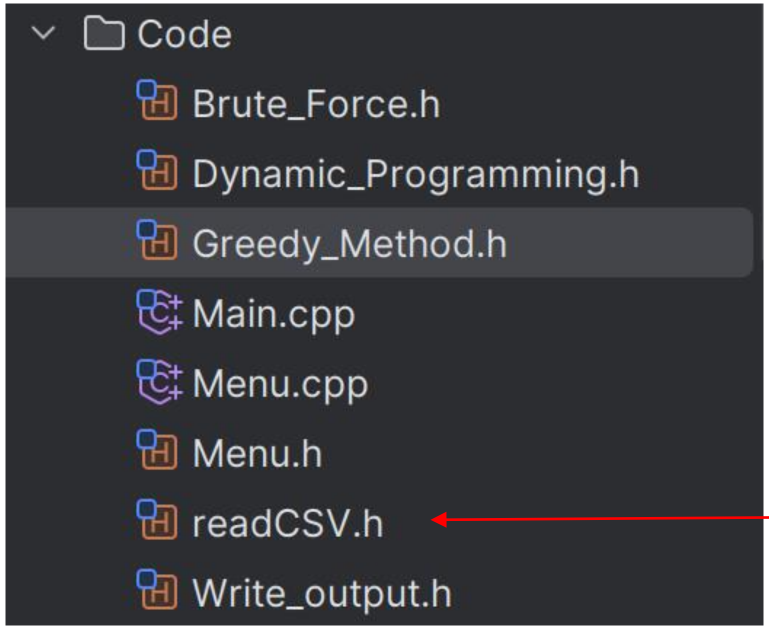
This file is reponsible to write the final output of our algorithms. Which items were used, the total weight and profit, and the execution time

This file is responsible to read and parse the given CSV files (TruckAndPallets_xx.csv and Palletes_xx.csv), and store the given data in appropriate variables

Processes to read the given datasets

Reading the CSV files

- In order to read the given data files (the CSV files) we implemented that logic in readCSV.h.



In the next slide we will further explain what this file exactly does!

The logic behind the readCSV.h

- This file has 2 main functios implemented.
- LoadTruck() has parameters parsedCapacity, parsedPallets and filename. This function will read the capacity and number of pallets from a TruckAndPallets_xx.csv file. This function checks if the capacity and pallets columns are not empty and if their data is numeric. If so, it prints to the screen what was read.
- LoadPallets() has a similar logic, it has a vector of weights, profits and a string filename. This function saves all the weights and profits read in their appropriate vector. It uses similar logic as in LoadTruck to check if the data is numeric and is not empty, it also checks for tabs, white-spaces and end of line characters.

Algorithms

- Brute Force (normal version):

This algorithm tests every possible combination of items (pallets) to find the one combination with with the highest profit that does not exceed the weight limit.

For n items, there are 2 ^ n possible subsets, for each subset we calculate the total weight and profit. If the total weight is within the allowed maximum and the profit is higher than any previously found, store this as best solution. At the end, return the highest profit found and the corresponding items.

Time Complexity: $O(2^n)$;

Space Complexity: O(n);

- Greedy Algorithm (profit-to-weight solution):

This function implements a greedy heuristic for the problem, selecting items based on the highest profit-to-weight ratio. Initialize all items as unused and set total profit and used weight to 0.

In each iteration, find the unused item with the highest profit-to-ratio that fits the remaining capacity. If multiple items have the same ratio, we select the one with lower weight. We then add the selected item to the result, update the total profit, used weight, and remaining capacity.

We repeat this until no more items can be added.

Time Complexity: $O(n^2)$;

Space Complexity: O(n);

Algorithms Continuation 1

- Greedy (Profit version):

This algorithm selects items based on the highest profit.

We initialize all items as unused, and set total profit and used weight to zero. In each iteration, find the unused item with the highest profit that fits the remaining capacity. If there is a tie, pick the one with the lower weight. Add the selected item to the result, update the total profit, used weight, and remaining capacity. Repeat until no items can be added.

Time Complexity: $O(n^2)$;

Space Complexity: $O(n)$;

- Greedy (optimal solution):

This algorithm runs both greedy strategies – by profit and by profit-to-weight ratio, on the same input. It then compares their results and returns the better one (higher profit), along with the corresponding items and total weight used.

Time Complexity: $O(n^2)$;

Space Complexity: $O(n)$;

Algorithms Continuation 2

- Dynamic-Programming:

The algorithm finds the optimal combination of pallets to include in a truck with limited weight capacity, maximizing the total profit.

It consists of three main steps:

Step 1. Initialize Base Cases: Creates a 2D table maxValue[item][weight]. For the first item, fills the table: if weight capacity >= item's weight, include its profit; otherwise, profit = 0.

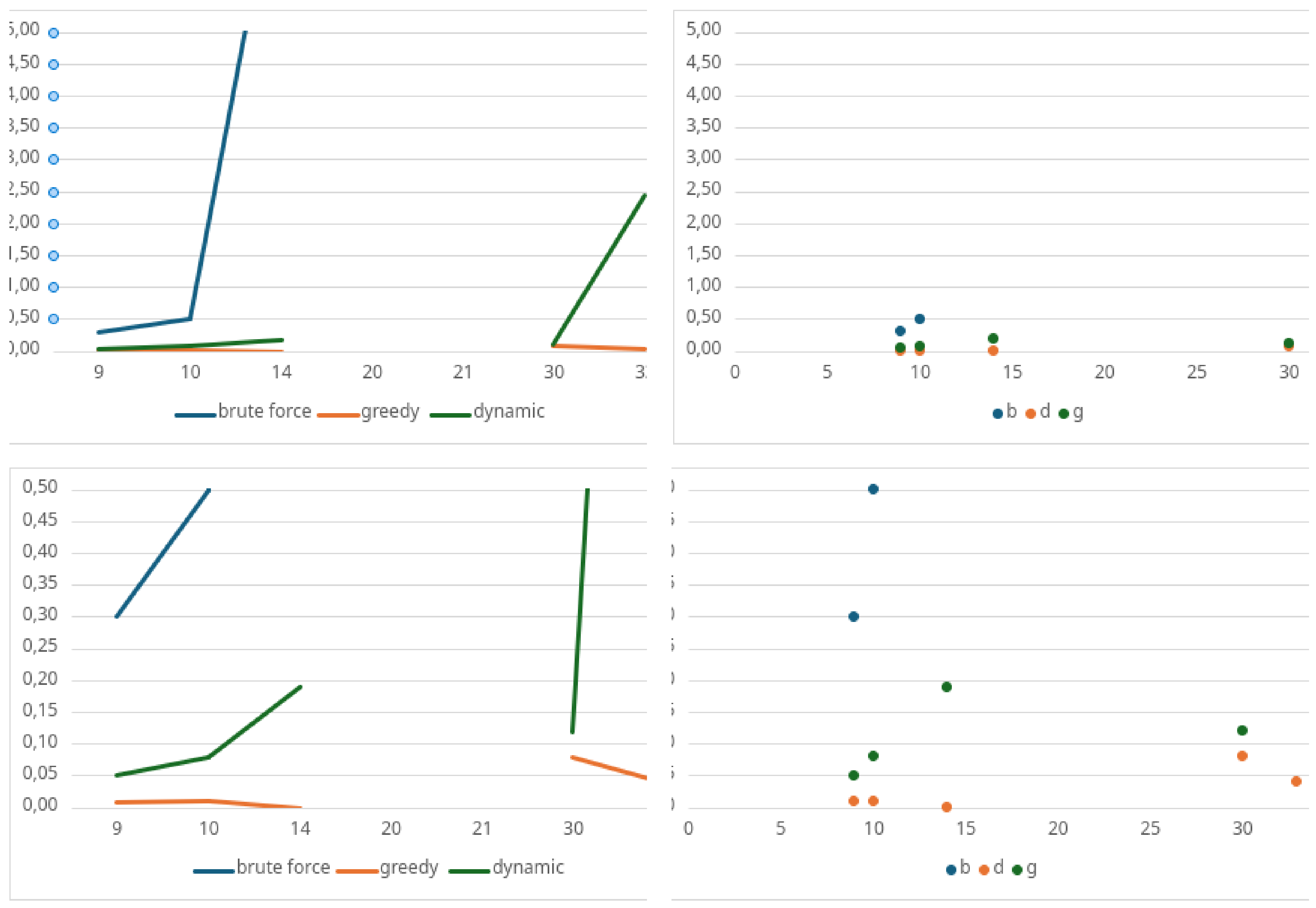
Step 2. Fill DP Table: For each item i and weight capacity k: If item's weight > capacity: take previous best value (can't include this item) Otherwise: choose maximum between: Not including current item: maxValue[i-1][k] or Including current item: maxValue[i-1][k-weights[i]] + profits[i] Step 3. Backtrack to Find Solution: Starting from maxValue[n-1][maxWeight], traces back which items were selected. Works backwards through items, checking if including each item was optimal.

Builds the usedItems boolean vector and calculates usedWeight. The algorithm guarantees finding the optimal solution by systematically considering all possible combinations through the DP table, avoiding redundant calculations. The main goal is that optimal solutions to subproblems can be combined to solve larger problems optimally.

Time Complexity: O(n × maxWeight);

Space Complexity: O(n × maxWeight);

Graphics to Compare Execution Times

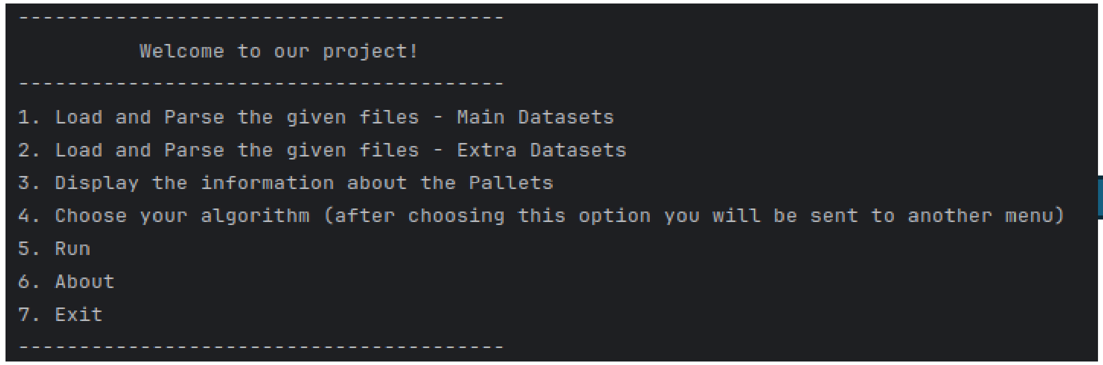


User Interface

- Our user interface has a Main Menu with the following options:
  - Load and Parse the CSV files;
  - Display the information about each dataset;
  - Choosing which algorithm to run;
  - The About Menu;
  - The Run option;
  - And the Exit option, which closes the program.

Example Usage

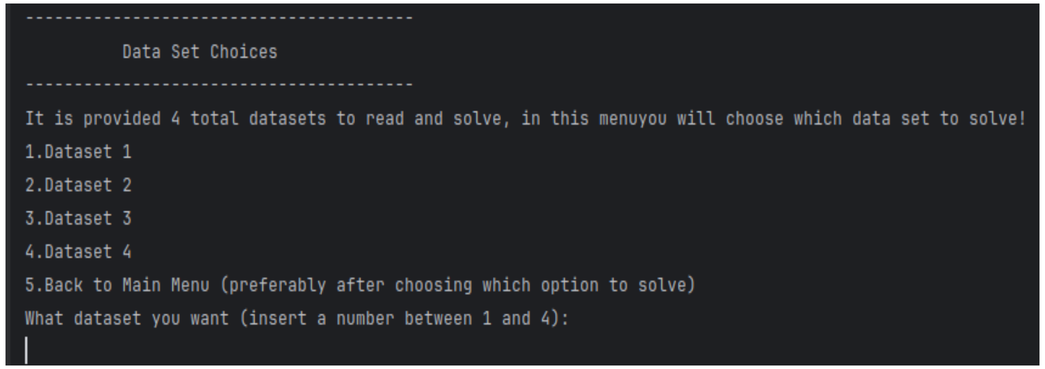
Loads and Parses the CSV files



```
----------------------------------------
         Welcome to our project!
----------------------------------------
1. Load and Parse the given files - Main Datasets
2. Load and Parse the given files - Extra Datasets
3. Display the information about the Pallets
4. Choose your algorithm (after choosing this option you will be sent to another menu)
5. Run
6. About
7. Exit
----------------------------------------
```

Pressing 1 or 2

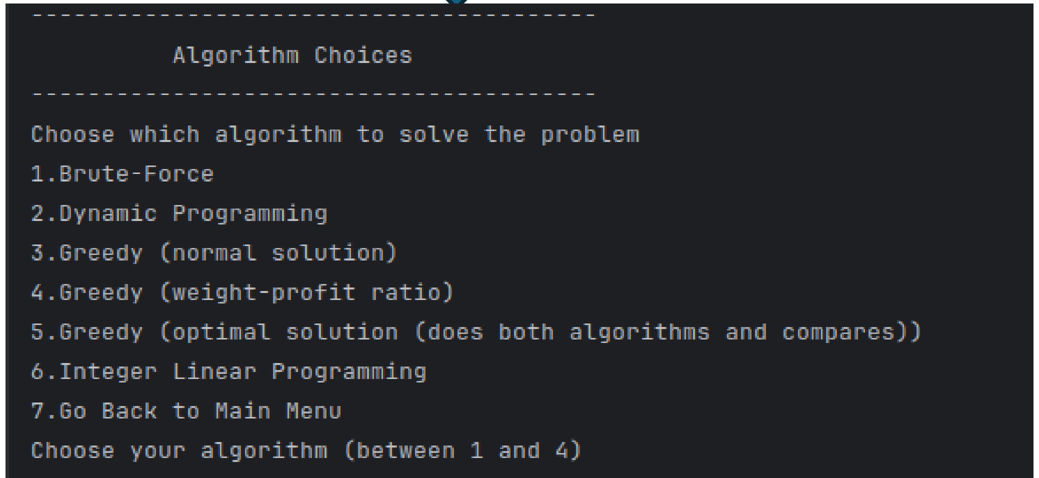
Choose Dataset and go back to Main Menu



```
----------------------------------------
          Data Set Choices
----------------------------------------
It is provided 4 total datasets to read and solve, in this menuyou will choose which data set to solve!
1.Dataset 1
2.Dataset 2
3.Dataset 3
4.Dataset 4
5.Back to Main Menu (preferably after choosing which option to solve)
What dataset you want (insert a number between 1 and 4):
```

Pressing 4 in the Main Menu then choose the desired Algorithm

It asks for the algorithm

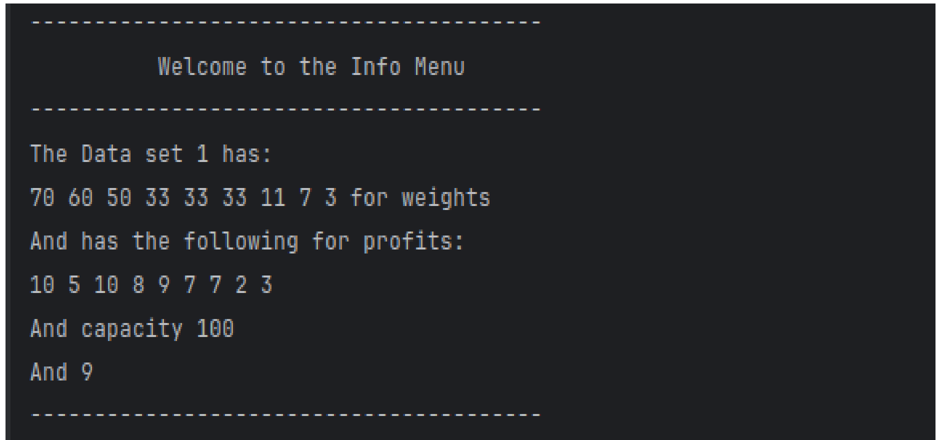


```
----------------------------------------
          Algorithm Choices
----------------------------------------
Choose which algorithm to solve the problem
1.Brute-Force
2.Dynamic Programming
3.Greedy (normal solution)
4.Greedy (weight-profit ratio)
5.Greedy (optimal solution (does both algorithms and compares))
6.Integer Linear Programming
7.Go Back to Main Menu
Choose your algorithm (between 1 and 4)
```

Optional command

Pressing 7 to go to Menu and then pressing 3

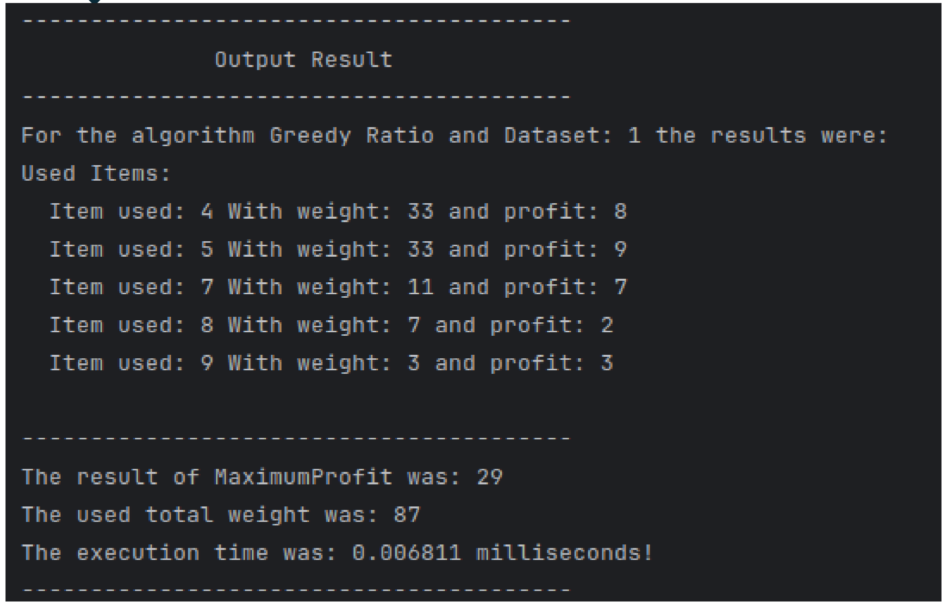
Display information about the chosen dataset



```
----------------------------------------
         Welcome to the Info Menu
----------------------------------------
The Data set 1 has:
70 60 50 33 33 33 11 7 3 for weights
And has the following for profits:
10 5 10 8 9 7 7 2 3
And capacity 100
And 9
----------------------------------------
```

Main Menu Press 5

Example Usage Continuation



```
----------------------------------------
              Output Result
----------------------------------------
For the algorithm Greedy Ratio and Dataset: 1 the results were:
Used Items:
  Item used: 4 With weight: 33 and profit: 8
  Item used: 5 With weight: 33 and profit: 9
  Item used: 7 With weight: 11 and profit: 7
  Item used: 8 With weight: 7 and profit: 2
  Item used: 9 With weight: 3 and profit: 3

----------------------------------------
The result of MaximumProfit was: 29
The used total weight was: 87
The execution time was: 0.006811 milliseconds!
----------------------------------------
```

Choosing the option Run, will print to the screen the output of the chosen algorithm, using the desired dataset. It displays which items were used, the weight and the profit of each one.

The final output section shows the result profit, the total weight used and the execution time.

Functionalities to highlight:

We’re proud to have added 2 new strategies to our greedy implementation, and we believe these extra features make our project even more unique.

Participation of each member

- Gonçalo: 33%
- Gustavo: 33%
- Tomás: 33%