**UMT**

**Advanced Algorithms and Programming**

**Quantifying greedy strategies’ efficiencies by statistical validation procedure.**

**Worked by: Eljon Zagradi Professor: Rene Natowicz**

*I conducted testing with 2000 instances for each algorithm in my program. This large sample size helped me analyze the performance differences between the greedy and dynamic programming approaches comprehensively.*

*For each of the problems I have provided “randomArray” function which generates a random array of integers 1D or 2D depending on the Problem. Also I provided the “getR” which calculates the relative distace of greedy and dynamic programming in (%). If needed also the display functions are provided.*

*For each Problem is provided a program that tests and compares the performance of the dynamic programming and greedy algorithms. The program conducts 2000 iterations, generating random instances of the problem and measuring the execution times for both algorithms. It calculates the maximum -> (MVB, MSM, TB) or minimum -> (MCP) value obtained by each algorithm and collects the results. Finally, the results are visualized in a histogram, providing insights into the performance differences between the two approaches. The code demonstrates a systematic approach to evaluating and comparing algorithmic solutions for all the problems.*

*The printouts for each problem are saved at respective files:*

*Maxiimum Value Bag -> MVB.txt*

*Maximum Sum of Marks -> MSM.txt*

*Minimum Cost Path -> MCP.txt*

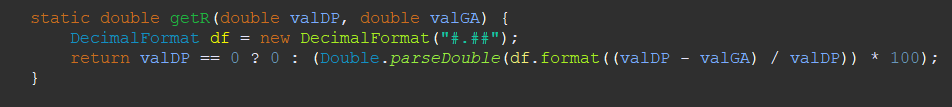
*Two Bags - > TB.txt*

*The full code for this Project is provided in this email but also can be found here: https://github.com/eljonzagradi/UMT\_Eljon\_Zagradi*

1. Maximum Value Bag Algorithm

A screen shot of a computer code

Description automatically generated with low confidence



A screen shot of a computer program

Description automatically generated with low confidence

A screen shot of a computer program

Description automatically generated with low confidence

A screen shot of a computer code

Description automatically generated with low confidence

For the greedy approach, I implemented a class called "Item" to facilitate the storage of an item's value and size.

The provided code demonstrates the implementation of the greedy algorithm for computing the maximum value in a bag. Here's a breakdown of how the algorithm works:

* The algorithm creates a list called "items" to store the items, each having a value and a size.
* It iterates over the value and size arrays to create item objects and adds them to the "items" list.
* The items in the list are sorted based on their value ratio in descending order.
* The algorithm initializes variables to keep track of the total value and remaining capacity of the bag.
* It performs a greedy selection of items by iterating over the sorted list and checking if an item can fit in the remaining capacity.
* If an item can fit, its value is added to the total value, and the item's size is subtracted from the remaining capacity.
* A picture containing text, screenshot, software

  Description automatically generatedFinally, the algorithm returns the total value, representing the maximum value that can be achieved using

Also attached to this file you will find the MVB.txt file which contains the printouts of the values generated as shown in the main class. Also contains the histogram generated by java code.

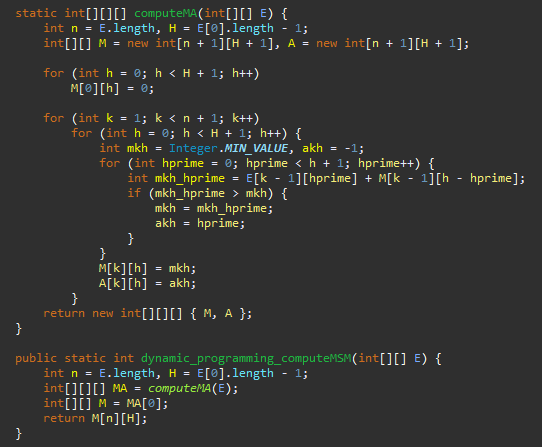
1. Maximum Sum of Marks:

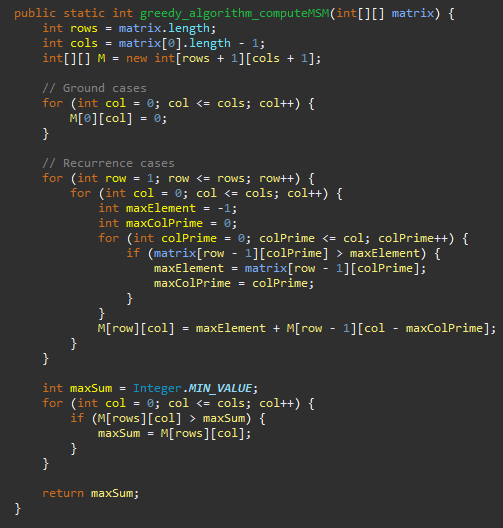
A screen shot of a computer program

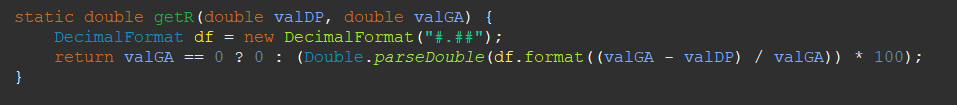
Description automatically generated with low confidenceA picture containing text, screenshot, font

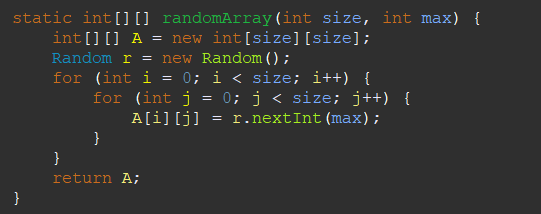
Description automatically generatedA picture containing screenshot, text, font

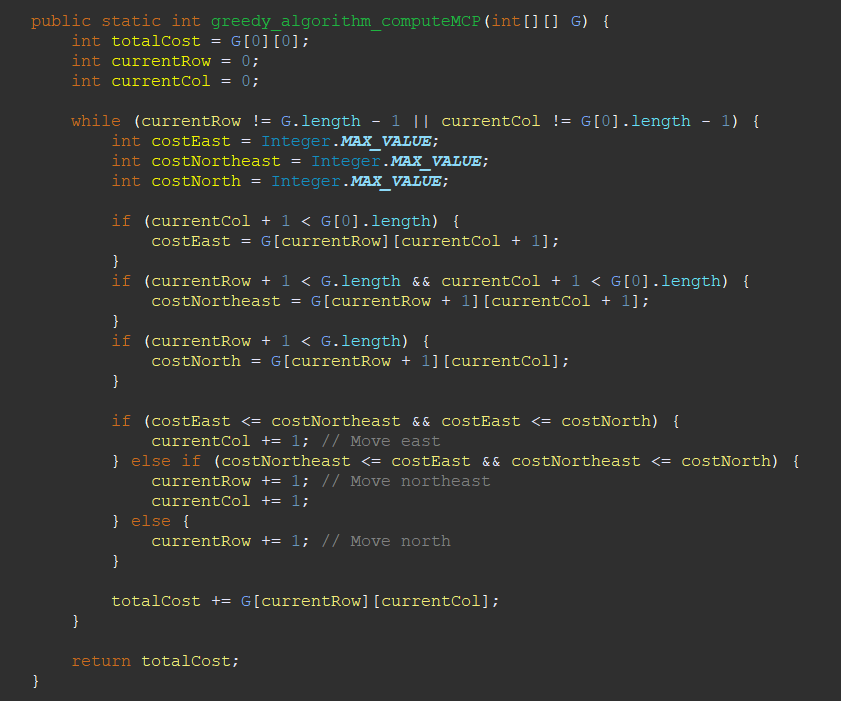
Description automatically generated





1. A picture containing text, screenshot, font

   Description automatically generatedMinimum Cost Path

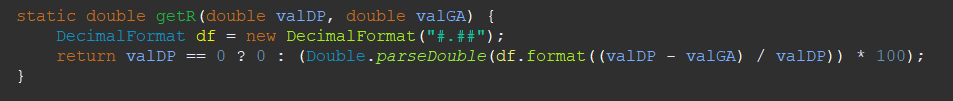
A picture containing text, screenshot

Description automatically generated

1. A screen shot of a computer

   Description automatically generated with medium confidenceA picture containing text, screenshot, font

   Description automatically generatedA screen shot of a computer program

   Description automatically generated with low confidenceTwo Bags

A screen shot of a computer program

Description automatically generated with low confidence

A screen shot of a computer program

Description automatically generated with low confidence

A screen shot of a computer code

Description automatically generated with low confidence

A screen shot of a computer code

Description automatically generated with low confidence