lab_report_knapsack

```r
library(AdvRknapsack)
```

Introduction

This vignette provides a comprehensive overview of the `AdvRknapsack` package, which offers solutions to the classic knapsack problem using three different algorithms: Brute-Force, Greedy, and Dynamic Programming. Each approach has its own strengths and trade-offs, depending on the size and nature of the input data.

Brute-Force Approach

The brute-force approach attempts every possible combination of items to find the optimal solution. While this guarantees the best solution, it is computationally expensive, especially for larger datasets.

Usage

```r
# Prepare Data
set.seed(42)
knapsack_objects <- data.frame(w = sample(1:4000, size = 2000, replace = TRUE), v = runif(n = 2000, 0,

# Run Brute-Force Knapsack
result <- brute_force_knapsack(x = knapsack_objects[1:16, ], W = 2000)
print(result)
#> $value
#> [1] 15880.86
#>
#> $elements
#> [1] 10 13 15
```

Greedy Approach

The greedy algorithm sorts items based on their value-to-weight ratio and picks the best option. While fast, it does not always yield the optimal solution.

Usage

```r
# Run Greedy Knapsack
result <- greedy_knapsack(x = knapsack_objects[1:100, ], W = 2000)
print(result)
```

```
#> $value
#> [1] 51593.09
#>
#> $elements
#>  [1] 89 75 63 77 83 33 15 98 35 60
```

Dynamic Programming Approach

Dynamic programming breaks the problem into smaller sub-problems and builds up the solution iteratively. It guarantees an optimal solution with improved efficiency over brute-force.

Usage

```
# Run Dynamic Knapsack
result <- knapsack_dynamic(x = knapsack_objects[1:500, ], W = 2000)
print(result)
#> $value
#> [1] 101868.4
#>
#> $elements
#>  [1]  63  75  77  83  89 110 187 203 219 244 256 283 320 322 329 341 455 495
```

Comparing the Methods

The choice of algorithm depends on the problem size and requirements:

- **Brute-Force**: Suitable for small datasets where exact results are required.
- **Greedy**: Good for larger datasets when a near-optimal solution suffices.
- **Dynamic Programming**: Ideal for medium-sized datasets where an optimal solution is desired with reasonable computational effort.

Conclusion

The `AdvRknapsack` package provides flexible options for solving the knapsack problem using different algorithms, catering to various performance needs and dataset sizes.