Knapsack Problem Using Greedy Technique

This project explores the use of a greedy algorithm to solve the knapsack problem, a classic optimization problem with real-world applications. The goal is to maximize the total value of items placed in a knapsack, while respecting the weight capacity constraints. The proposed approach leverages the greedy technique to construct an efficient solution, ensuring the formation of the maximum number of groups with strictly increasing lengths.

NY

0-1 Knapsack Problem

Formal description: Given two n-tuples of positive numbers

$$\langle v_1, v_2, \ldots, v_n \rangle \quad \text{and} \quad \langle w_1, w_2 \ldots, w_n \rangle,$$

and $W > 0$, we wish to determine the subset $T \subseteq \{1, 2, \ldots, n\}$ (of files to store) that

$$\text{maximizes} \quad \sum_{i \in T} v_i,$$

$$\text{subject to} \quad \sum_{i \in T} w_i \leq W.$$

Knapsack Problem Using Greedy Technique

Certified Work

This Capstone project report, "Knapsack Problem Using Greedy Technique For Real Time Applications and finding Maximum Number of Groups With Increasing Length", is the bonafide work of the student Y. Nanda Kishore Reddy (192211435).

Supervisor Approval

The project work was carried out under the supervision of Dr. R Dhanalakshmi, Professor in the Department of Machine Learning at SIMATS Engineering, Saveetha Institute of Medical and Technical Sciences.

Department Approval

The report has been reviewed and approved by Dr. S. Mehaboob Basha, Head of the Department of Machine Learning at SIMATS Engineering, Saveetha Institute of Medical and Technical Sciences.

Problem Statement

Usage Limits

A 0-indexed array "usageLimits" of length n is provided, where each element represents the maximum number of times the corresponding number can be used in the groups.

Output

The algorithm should return an integer representing the maximum number of groups that can be formed while meeting the given requirements.

1

2

3

Group Formation

The goal is to form the maximum number of groups using the numbers 0 through n-1, ensuring that no number is used more than its specified limit and that each group (except the first) is strictly longer than the previous one.

Greedy Algorithm Approach

Sorting

The first step is to sort the usage limits array in ascending order to make it easier to form groups, starting with the numbers that have the lowest usage constraints.

Usage Tracking

A list called "usage_counts" is maintained to keep track of how often each number has been used, ensuring that the usage limits are not exceeded.

Group Formation

The algorithm iteratively determines the largest possible group size that can be formed while respecting the usage constraints. The group count is incremented, and the usage counts are updated accordingly.

Detailed Example

1

Input

For the input "usageLimits = [1, 2, 5]", the algorithm will calculate the maximum number of groups that can be formed.

2

Group Sizes

The algorithm computes the possible group sizes: S1 = 1, S2 = 3, S3 = 6, and S4 = 10. Since the total usage capacity of 8 is enough for groups up to size 3, the maximum number of groups is 3.

3

Group Formation

The groups can be formed as: Group 1 with 1 element, Group 2 with 2 elements, and Group 3 with 3 elements, respecting the usage limits.

Time Complexity

1 Sorting

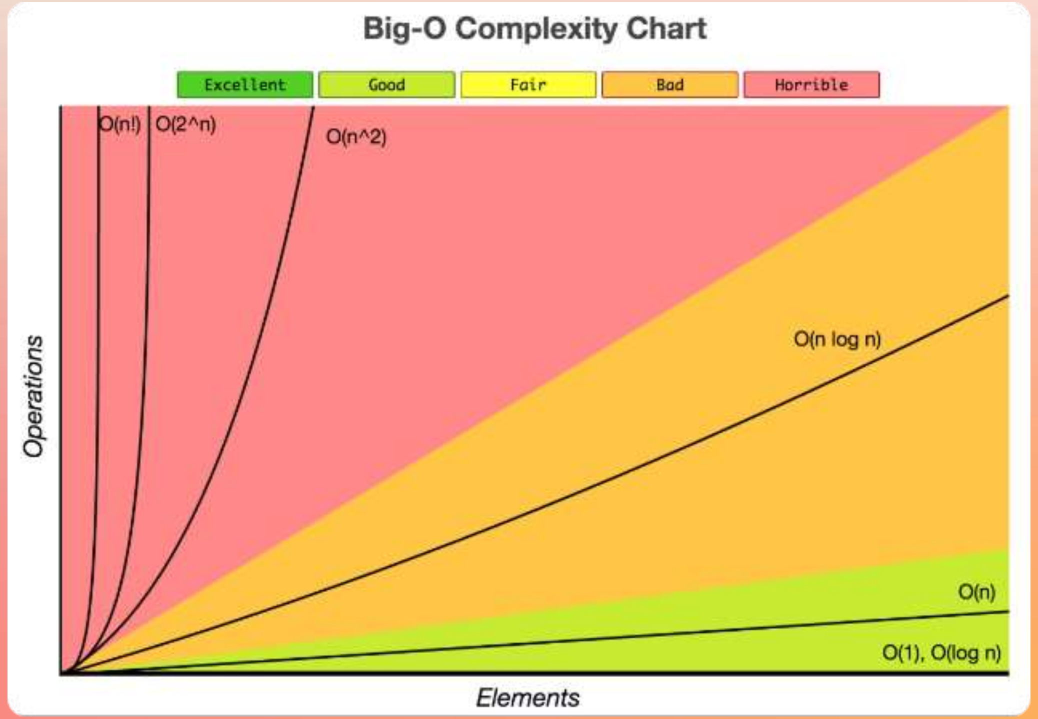
The time complexity of sorting the usage limits array is O(n log n).

2 Group Formation

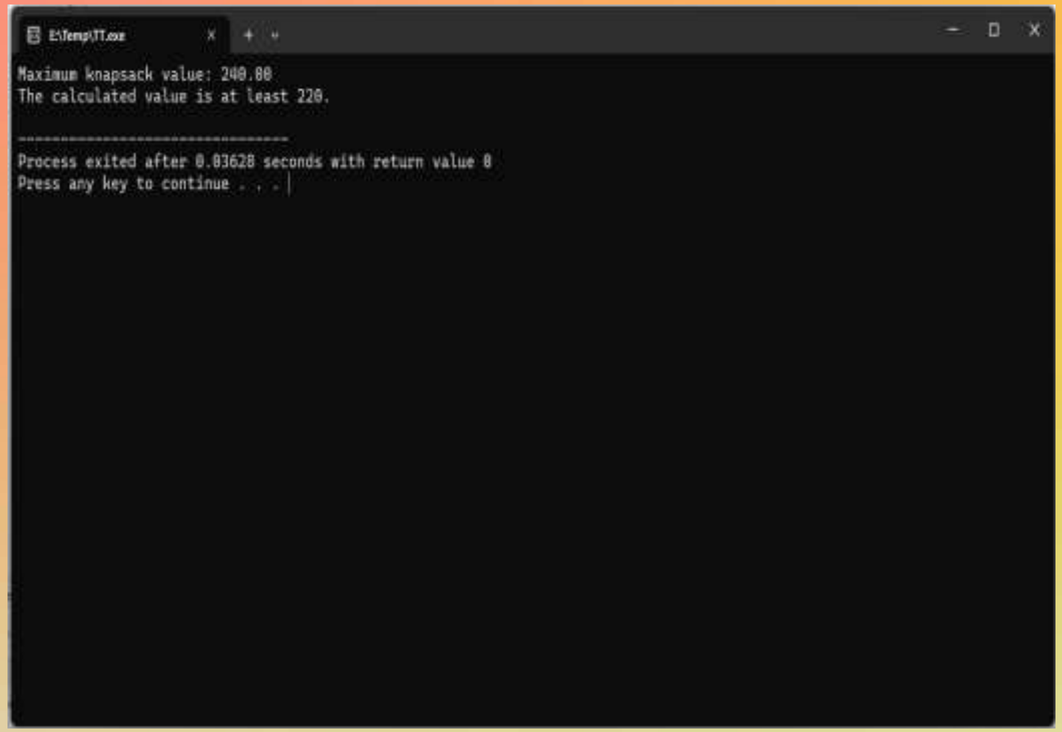
The loop that determines the group sizes and updates the usage counts has a time complexity of O(n).

3 Overall Complexity

The overall time complexity of the greedy algorithm is O(n log n), which is efficient for solving the problem.



OUTPUT :



Real-World Applications

Resource Allocation

The knapsack problem and its greedy solution can be applied to optimize resource allocation in various domains, such as logistics, project management, and investment planning.

Scheduling

The algorithm can be used to schedule tasks or activities with varying resource requirements, ensuring the maximum utilization of available resources.

Portfolio Optimization

In finance, the greedy approach can be employed to construct optimal investment portfolios, maximizing the overall value while respecting budget constraints.

Conclusion

Efficient Solution

The proposed greedy algorithm provides an efficient and effective solution to the problem of forming the maximum number of groups with strictly increasing lengths, while respecting the given usage constraints.

Practical Applications

The algorithm's ability to solve the knapsack problem can be leveraged in various real-world scenarios, such as resource allocation, scheduling, and portfolio optimization, making it a valuable tool for optimization and decision-making.

Future Enhancements

Further research and development could explore ways to extend the algorithm's capabilities, such as handling dynamic usage constraints or incorporating additional optimization criteria, to broaden its applicability in diverse domains.

Acknowledgments

Supervisor

The authors would like to express their gratitude to Dr. R Dhanalakshmi, Professor in the Department of Machine Learning at SIMATS Engineering, for her invaluable guidance and support throughout the project.

Department

The authors also acknowledge the contributions and feedback provided by Dr. S. Mehaboob Basha, Head of the Department of Machine Learning at SIMATS Engineering, Saveetha Institute of Medical and Technical Sciences.

Institution

The authors would like to thank the Saveetha Institute of Medical and Technical Sciences for providing the necessary resources and infrastructure to conduct this research project.