GREEDY ALGORITHM : A STUDY

Mr. Juned Khatri

Research Scholar, Department of Mathematics, Sage University, Indore
Email id- junedkhatrijnt@gmail.com &
Dr. Aarti Sharma Associate Professor Department of Mathematics, Sage University, Indore
Email id- sharma.aarti@sageuniversity.in

ABSTRACT

The greedy algorithm is a simple optimization technique that makes the locally optimal choice at each step in the hope of finding a globally optimal solution. The greedy algorithm is a commonly used algorithm design idea that can provide efficient solutions to many practical problems. Greedy algorithm is an algorithmic paradigm that follows the problem-solving approach of making the locally optimal choice at each stage with the hope of finding the global optimum greedy algorithm is often used to solve the problems of some decisions. This review will introduce in detail the basics of greedy algorithms, summarize the practical experience of the application of greedy algorithms, provide guidance and suggestions for solving problems using greedy algorithms in various field with its advantages and limitations.

Keywords:

Greedy Algorithm, The Principle of the Greedy Algorithms, Optimal Solution, Application of Greedy Algorithms in the Real-World Advantages and Limitations

INTRODUCTION

The Greedy algorithm was proposed in 1959 by Dutch computer scientist and mathematician Edsger W Dijkstra to solve the unit shortest path problem. He developed the concept while working on finding the minimum spanning tree in a graph. The core idea of a greedy algorithm is to make the best possible choice at each step, without considering the overall long-term consequences, in the hope of reaching a globally optimal solution. The main advantages of the greedy algorithms are simplicity, efficiency, and case of implementation. It typically does not require sorting input data or calculating complex data structures, so it has a low time complexity. [1]

Greedy algorithms are usually easy to think of, easy to implement and run fast. Proving their correctness may require rigorous mathematical proofs and is sometimes insidious hard. In addition, greedy algorithms are infamous for being tricky. Missing even a very small detail can be fatal. But when you have nothing else at your disposal, they may be the only salvation. With backtracking or dynamic programming, you are on a relatively safe ground. With greedy instead, it is more like walking on a mined field. Everything looks fine on the surface but the hidden part may backfire on you when you least expect. While there are some standardized problems, most of the problems solvable by this method call for heuristics. There is no general template on how to apply the greedy method to a given problem; however the problem specification might give you a good insight. In some cases, there are a lot of greedy assumptions one can make, but only few of them are correct. They can provide excellent challenge opportunities. [2]

Greedy design technique is primarily used in optimization problems. Optimization problems are problems wherein we would like to find the best of all possible solutions. In other words, we need to find the solution which has the optimal (maximum or minimum) value satisfying the given constraints. The Greedy approach helps in constructing a solution for a problem through a sequence of steps where each step is a partial solution. This partial solution is extended progressively to get the complete solution. In the greedy approach each step chosen must satisfy the constraints given in the problem. Each step is chosen such that it is the best alternative among all feasible choices that are available. The choice of a step once made cannot be changed in subsequent steps**.** [3]

Greedy algorithms can be characterized as being 'short sighted', and as 'non-recoverable'. They are ideal only for problems which have 'optimal substructure'. Despite this, greedy algorithms are best suited for simple problems (e.g. giving change). It is important, however, to note that the greedy algorithm can be used as a selection algorithm to prioritize options within a search, or branch and bound algorithm.

There are a few variations to the greedy algorithm: (a) Pure greedy algorithms (b) Orthogonal greedy algorithms (c) Relaxed greedy algorithms. [4]

CHARACTERISTICS OF GREEDY ALGORITHM

- Greedy algorithms are simple, straightforward, and easy to implement.
- They are efficient in terms of time complexity, often providing quick solutions.
- They take decisions based on information at hand without worrying about the effect these decisions may have in the future.
- They work in stages and never reconsider any decision.
- These algorithms do not reconsider previous choices, as they make decisions based on current information without looking ahead. [5]

THE PRINCIPLE OF THE GREEDY ALGORITHMS

1 Greedy choice nature

The nature of greedy selection or choice means that when faced with a problem and needs to decide, the greedy algorithm will give preference to the solution that currently seems optimal, regardless of the global optimal solution. The key to greedy choice is that the choice at each step must be locally optimal, even if this choice may result in the result not necessarily being globally optimal. Greedy selection is generally suitable for satisfying problems with specific properties that have optimal substructure properties and that produce optimal solutions. The optimal substructure property means that the overall optimal solution to the problem can be achieved by a series of local optimal choices. An important feature of greedy selection is that it does not backtrack, i.e. once a choice is made, it does not change. This is because greedy algorithms are usually based on local optimal selection, focusing only on the optimal solution of the current step, and do not consider future choices and influences**.** [6]

2 Optimal Substructures

An optimal substructure is a problem- solving idea that is commonly used in dynamic programming algorithms and greedy algorithms. An optimal substructure is one in which the optimal solution of a problem contains the optimal solution of its sub-problems. Specifically, the property that a problem has an optimal substructure means that the optimal solution of the sub problem. That is, the optimal solution of the problem can be obtained solution to a sub problem and then doing further operations according to some rules or algorithms. The nature of optimal substructure enables users to solve problems more efficiently. When a user faces a complex problem, it can be divided into multiple interrelated sub-problems. By solving the sub problem and taking advantage of the properties of the optimal solution of the entire problem. The optimal substructure is an important problem-solving idea that provides us with a way to disassemble and solve complex problems. By rationally designing sub problem solving strategies and taking advantage of the properties of optimal substructures, we can efficiently solve many practical problems. [7]

ADVANTAGES

Significant advantages of greedy algorithms are as follows

(1). Simple and easy to understand: The greedy algorithm builds the optimal solution to the whole based on the optimal selection of each step, and the logic is relatively simple, easy to understand and implement.

(2). High efficiency: Since the greedy algorithm only focuses on the optimal solution of the current step and does not consider the global calculation, it can quickly find a good approximate solution to some problems.

(3). Can solve some optimization problems: For some specific types of problems, greedy algorithms can find global optimal solutions, such as some backpack problems, graph theory problems, etc.

(4). Can be used as an optimization strategy for other algorithms. The greedy algorithm can be used as an optimization strategy or auxiliary algorithm for other algorithms to improve the efficiency of other algorithms through the selection of greedy strategies. [8]

LIMITATIONS

The greedy algorithm has the following limitations:

(1). No regret mechanism: Greedy algorithms usually have no mechanism for fallback, and once a choice is made, it cannot be undone. This means that once a wrong choice is made, it may not be corrected, resulting in a distortion of the final choice result.

(2). Need to satisfy the greedy selection nature: The greedy algorithm requires the problem to have the nature of greedy selection, that is, the global optimal solution can be obtained through local greedy selection. However, not all problems have this nature, and greedy algorithms cannot get the right results for problems that cannot meet this nature.

(3). Need to be able to decompose sub problems: Greedy algorithms usually need to break down the original problem into several sub problems to solve. This means that the problem must have a nature that can be broken down into sub problems; otherwise, greedy algorithms cannot be used.

In summary, the greedy algorithm can get better results on some problems, but for complex problems with many constraints, the limitations of the greedy algorithm will lead to its inability to get the correct results. In practical applications, it is necessary to carefully choose whether to use greedy algorithms based on the characteristics and requirements of the problem. [9]

OPTIMAL SOLUTION

The greedy algorithm can be optimized in combination with the following algorithms:

(1). Divide and conquer algorithm: The greedy algorithm can be optimized since the divide and conquer algorithm, and the appropriate greedy strategy can be selected to make the sub-problem of divide and conquer easier to solve.

(2). Dynamic programming: The greedy algorithm can find the candidate solution of the optimal solution in the process of dynamic programming, thereby reducing the state space and computation amount of dynamic programming.

(3). Backtracking algorithm: The greedy algorithm can be used as a heuristic search strategy for the backtracking algorithm, and the most likely next step is selected according to the greedy criterion, thereby improving the efficiency of the backtracking algorithm. [10]

APPLICATION OF GREEDY ALGORITHMS IN THE REAL WORLD

Greedy algorithms have vast applicability in real-life scenarios. Greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data. They can make commitments to certain choices too early which prevent them from finding the best overall solution later. For example, all known greedy coloring algorithms for the graph coloring problem and all other NP-complete problems do not consistently find optimum solutions. Nevertheless, they are useful because they are quick to think up and often give good approximations to the optimum. If a greedy algorithm can be proven to yield the global optimum for a given problem class, it typically becomes the method of choice because it is faster than other optimization methods like dynamic programming. Examples of such greedy algorithms are Kruskal's algorithm and Prim's algorithm for finding minimum spanning trees, Dijkstra's algorithm for finding single-source shortest paths, and the algorithm for finding optimum Huffman trees. The distribution tasks that involve rationed resources that must be distributed among several competing needs are usually performed using greedy strategies. Greedy optimization is used rather pervasively in real-life applications; for example, data compression. One of the most basic compression methods is Huffman coding, which generates efficient prefix codes for symbols in relation to the frequency by a greedy algorithm. Chronologically assigning short code words to frequent symbols reduces the overall storage space needed. Machine learning algorithms also appear with greedy thinking incorporated into their systems. The densification technique of decision tree learning continually picks the attribute in each loop that creates the "purest" divisions of subgroups and goes on subdividing the observations in that way until their conclusion or end branches in a prediction. This greedy construction of optimal splitting decisions constructs decision trees step by step and therefore improves interpretability. [11]

These are to illustrate how greedy algorithms may be applied to various aspects of resource storage and retrieval, and information sorting and assessment. The efficiency of their methods and reasonable solutions derived from iterative improvement of solutions within local optima allow for solving many problematic issues of optimization maintenance in computationally intensive practical applications. When a global optimum cannot be determined, greedy algorithms' blend of strong throughput with decent solution quality allows for a wide range of application areas for system management. Due to their practical, straightforward, and efficient characteristics, these are some of the most useful techniques for simulating optimization

problems in technology, science, and other fields where optimal solutions may not necessarily be necessary on a global scale. [12]

CONCLUSION

The greedy algorithms, have become firmly rooted in the field of computer science because of the demonstrated ability of generating solutions that are approximations of the optimal values rapidly. These basic advantages stem from the simplicity of ideas that underlie them, time requirements, and the ability to iteratively choose the locally optimum decision. Even though greedy heuristics do not always provide global optimality, they are used because they always provide reasonable, approximate, or optimal solution. This makes it possible to solve very large problem instances intractable to thin, more general exhaustive approaches. When the problems belong to well-defined categories that can be mapped to greedy principles, we know that better solutions are guaranteed to bolster this strategy's effectiveness. However, increasing compatibility with greedy algorithm properties, such as optimal substructure and the greedy choice property improves the solution.

REFERENCES

[1] Dijkstra, E. W. *A Note on Two Problems in Connexons with Graphs. Numeric Mathematics.* 1959, pg. 269–71.
[2] Https://Www.Topcoder.Com/Thrive/Articles/Greedy%20is%20Good. accessed 3 July, 2025
[3] Malik, Annu, et al. "Greedy Algorithm." *International Journal of Scientific and Research Publications*, no. ISSN 2250-3153, Aug. 2013, pg. 1.
[4] *Geeksforgeeks.Org/Dsa/Introduction-to-Greedy-Algorithm-Data-Structured-&-Algorithm-Tutorials/*. accessed 18 May, 2025.
[5] *Aecedu.in/Ace/Instruction Material/DAA%20u-2*. accessed 15 June, 2025
[6] Wen, L., et al. "Understanding the Limitation of Greedy Algorithms A Survey and Case Study." *IEEE ACCESS*, 2018.
[7] *Geeksforgeeks.Org/Dsa/Introduction-to-Greedy-Algorithm-Data-Structured-&-Algorithm-Tutorials/*. accessed 18 May, 2025.
[8] Zhang, Y., et al. "A Hybrid Approach Combining Greedy Algorithm and Divide-and-Conquer for Optimization Problems." *Journal of Experimental and Theoretical Artificial Intelligence*, 2018
[9] Wang, Yizhun. "Review on Greedy Algorithm." *Theoretical and Natura Science*, EWA Publishing, Nov. 2023, https//doi.org/10.54254/2753-8818/14/20241041.
[10] B, Smith. "A Hybrid Algorithm Combining Greedy and Backtracking Strategies for Optimal Scheduling Problem." *Journal of Optimization,* 2018, pg. 789–801.
[11] Malik, Annu, et al. "Greedy Algorithm." *International Journal of Scientific and Research Publications*, no. ISSN 2250-3153, Aug. 2013, pg. 1.
[12] Zhang, Jierui. *The Logic and Application of Greedy Algorithm*. 2024, https://doi.org/10.54254/2753-8818/14/20241041. accessed on 23 May, 2025