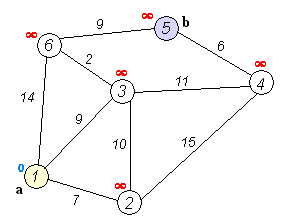
**Greedy Algorithms**

A **greedy algorithm** is a simple, intuitive algorithm that is used in optimization problems. The algorithm makes the optimal choice at each step as it attempts to find the overall optimal way to solve the entire problem. Greedy algorithms are quite successful in some problems, such as [Huffman encoding](https://brilliant.org/wiki/huffman-encoding/) which is used to compress data, or [Dijkstra's algorithm](https://brilliant.org/wiki/dijkstras-short-path-finder/), which is used to find the shortest path through a graph.



However, in many problems, a greedy strategy does not produce an optimal solution. For example, in the animation below, the greedy algorithm seeks to find the path with the largest sum. It does this by selecting the largest available number at each step. The greedy algorithm fails to find the largest sum, however, because it makes decisions based only on the information it has at any one step, without regard to the overall problem.

**Structure of a Greedy Algorithm**

Greedy algorithms take all of the data in a particular problem, and then set a rule for which elements to add to the solution at each step of the algorithm. In the animation above, the set of data is all of the numbers in the graph, and the rule was to select the largest number available at each level of the graph. The solution that the algorithm builds is the sum of all of those choices.

If both of the properties below are true, a greedy algorithm can be used to solve the problem.

* **Greedy choice property:** A global (overall) optimal solution can be reached by choosing the optimal choice at each step.
* **Optimal substructure:** A problem has an optimal substructure if an optimal solution to the entire problem contains the optimal solutions to the sub-problems.

In other words, greedy algorithms work on problems for which it is true that, at every step, there is a choice that is optimal for the problem up to that step, and after the last step, the algorithm produces the optimal solution of the complete problem.

To make a greedy algorithm, identify an optimal substructure or subproblem in the problem. Then, determine what the solution will include (for example, the largest sum, the shortest path, etc.). Create some sort of iterative way to go through all of the subproblems and build a solution.

## **Limitations of Greedy Algorithms**

Sometimes greedy algorithms fail to find the globally optimal solution because they do not consider all the data. The choice made by a greedy algorithm may depend on choices it has made so far, but it is not aware of future choices it could make.

Example

|  |
| --- |
| In the graph below, a greedy algorithm is trying to find the longest path through the graph (the number inside each node contributes to a total length). To do this, it selects the largest number at each step of the algorithm. With a quick visual inspection of the graph, it is clear that this algorithm will not arrive at the correct solution. What is the correct solution? Why is a greedy algorithm ill-suited for this problem? |

The correct solution for the longest path through the graph is 7, 3, 1, 997,3,1,99. This is clear to us because we can see that no other combination of nodes will come close to a sum of 9999, so whatever path we choose, we know it should have 9999 in the path. There is only one option that includes 9999: 7, 3, 1, 997,3,1,99.

The greedy algorithm fails to solve this problem because it makes decisions purely based on what the best answer at the time is: at each step it *did* choose the largest number. However, since there could be some huge number that the algorithm hasn't seen yet, it could end up selecting a path that does not include the huge number. The solutions to the subproblems for finding the largest sum or longest path do not necessarily appear in the solution to the total problem. The optimal substructure and greedy choice properties don't hold in this type of problem

## **Applications**

There are many applications of greedy algorithms.

**Dijkstra's Algorithm**

**Huffman Coding**

## **References**

1. , S. Greedy-search-path-example. Retrieved May 2, 2016, from <https://en.wikipedia.org/wiki/File:Greedy-search-path-example.gif>
2. , S. Greedy-search-path. Retrieved May 4, 2016, from <https://commons.wikimedia.org/wiki/File:Greedy-search-path.gif>
3. Okie , . Greedy Algorithms. Retrieved May 2, 2016, from <http://www.radford.edu/~nokie/classes/360/greedy.html>
4. , I. Dijkstra Animation. Retrieved May 2, 2016, from <https://commons.wikimedia.org/wiki/File:Dijkstra_Animation.gif>