Analysis and Application of Dynamic Programming

Jinyi Xu, Shaofei Wu*
School of Wuhan Institute of Technology University, Wuhan, China

*Corresponding author: 04005047@wit.edu.cn

Abstract. Dynamic programming is a part that is not well understood in the introductory learning of algorithms, but it is also a part worth learning. It has been successfully applied in many fields, such as gene sequencing, human flow control, and hydropower resource allocation. This article systematically explains the principle of dynamic programming. At the same time, through the comparison with other algorithms, we can deeply understand the nature of dynamic programming and its advantages and disadvantages in solving problems compared with other algorithms. Finally, it analyzes the problem-solving methods and steps of dynamic programming based on relevant application examples.

Keywords: Dynamic programming, memory recursion, knapsack problem.

1. Introduction
The dynamic programming method is a method to solve the multi-stage decision-making optimal solution problem. It is not as "dynamic" as the name suggests. In the process of solving a practical problem, it divides the big problem into small problems and defines the initial state. The current sub-problem can be solved by the previous sub-problem. The focus and difficulty of solving this problem is the connection between the current sub-problem and the previous sub-problem, that is, the state transition equation. After finding the state transition equation, the solution of the sub-problem is sought step by step from the bottom to the top of the initial state of the problem, so as to solve the final big problem.

2. Basic Idea of Dynamic Programming
First, determine whether the problem to be solved has optimal substructure properties, overlapping subproblems properties and no aftereffects, which determines whether the problem can be solved by dynamic programming. The optimal substructure means that the optimal solution of the problem contains the optimal solution of its sub-problems [1]; the overlapping sub-problem means that when the problem is decomposed into sub-problems, some of the sub-problems generated each time are repeated [1]; No aftereffect means that once the state of a certain stage is determined, it will not be affected by the subsequent decisions of this state [2].

The basic idea of using dynamic programming to solve problems is to divide a problem into multiple stages, and one stage has multiple states. From these states, the solution of this stage and the value of each state in the next stage can be obtained. And so on, until the solution of the last stage is found, that is, the solution of the problem.
In general, in the process of thinking about the problem, our thinking should be top-down. To solve the original problem, we need to solve the problem of the previous stage of the original problem, and there are multiple states in the previous stage. The choice of, all may constitute the solution of the original problem, which needs to be judged by the transfer equation, and these states are determined by the last stage... loop this process until the initial state. But in the process of calculation, its process is bottom-up. Starting from the initial state, the solution of each state of the first stage is calculated, and then the states of the next stage can be solved from these solutions until the solution of the last stage is completed.

3. Other related algorithms

3.1. Greedy method
In addition to dynamic programming, another very effective solution to the optimal problem is to adopt greedy ideas. But to use a greedy algorithm to solve the problem, this problem needs to satisfy the nature of greedy selection, that is, the overall optimal solution can be achieved through local optimal selection [3], which is more stringent than the application of dynamic programming. Most of the time, the problems that can be solved by the greedy algorithm can be solved by the dynamic programming method, but the problems that can be solved by the dynamic programming method may not be solved by the greedy method. It can be understood like this: Greedy is a special case of dynamic programming. Greedy only focuses on the present, while dynamic programming also looks back at history.

3.2. Divide and conquer
In essence, the dynamic programming algorithm is also a divide and conquer idea. Both of them divide a large problem into small problems and solve them one by one. The difference is that a sub-problem of the dynamic programming method may appear multiple times. The solution of the latter problem requires the solution of the previous sub-problem, that is, the sub-problems overlap. Therefore, we thought of storing these sub-problems so that when solving large sub-problems, we can directly obtain the solutions of small sub-problems, avoiding repeated calculations to obtain better algorithm efficiency; the divide-and-conquer method is more suitable for sub-problems that are independent. In the case of recursively solving the sub-problems one by one and then combining the sub-problems to solve the problem, it can also be used to solve the dynamic programming problem, but the algorithm efficiency will not be as high as the dynamic programming method. Therefore, the dynamic programming method can be understood as an algorithm idea based on the divide and conquer method.

3.3. Memory recursion
Similar to the dynamic programming method, the memory recursion is also used to solve the problem with the idea of space-for-time algorithm, and their essence is actually the same. But in general, the memory recursion is solved from the top down, while the dynamic programming method is bottom-up. The two are generally interchangeable. The dp table in dynamic programming is equivalent to the cache in memory recursion, and the state transition equation in dynamic programming is equivalent to recursive calling. The conversion between the two is similar to the conversion between recursion and loop to a certain extent.

4. Application

4.1. Problem solving steps
When we get a problem, we must first think about whether the problem can be solved by dynamic programming. If it can be solved by dynamic programming, then we have to think about whether the solution is optimal. As mentioned earlier, if the problem satisfies the optimal sub-structure, overlapping sub-problem properties and no aftereffect, then this problem can be solved by dynamic programming. If it cannot be solved by dynamic programming, then we will consider using other algorithms like greedy algorithms, divide and conquer algorithms, or memory recursion algorithms.
programming. Now we come to think about whether it is optimal to use dynamic programming to solve this problem. Suppose this problem has n stages, and each stage has m states. When m is equal to 1, this problem is suitable to be solved by recursion; if the optimal state of each stage is derived from the optimal state of the previous stage, then this problem is suitable to be solved by the greedy method; if each stage the optimal state of is derived from some state in a previous stage, then this problem is suitable to be solved by dynamic programming.

After determining that this problem is suitable for solving by dynamic programming, the problem is divided into multiple stages according to the characteristics of the problem. When the problem develops to a certain stage, we need to use different states to represent the objective situation of the problem at this time, and then What we need to find is the relationship between a certain state in a certain stage and the state of the previous stage, that is, to find the transition equation. Before that, we need to ensure that the choice of state has no aftereffect and find the initial state. Finally, find the optimal solution at each stage according to the transfer equation, and finally find the optimal solution at the final stage, that is, find the solution to the original problem.

4.2. Application examples
The 0-1 knapsack problem is a classic dynamic programming problem, and it is also a problem worth learning, because many problems can be transformed into a 0-1 knapsack problem to solve.

The description of the problem is as follows: There are n items with weights \(w_1, w_2, w_n\), and their values are \(v_1, v_2, v_n\), given a backpack with a capacity of \(W\). Design a plan to select some items from these items and put them into the backpack. Each item is either selected or not. It is required that the selected items can not only be placed in the backpack, but also have the greatest value.

First, it is natural to think of a very violent recursive solution. For each item, there are two possibilities, that is, to put it in the backpack or not to put it in the backpack. So, we can get the recurrence:

\[
f(n,W) = \max(f(n-1,W), f(n-1,W-w_n)+v_n)
\]

Among them, \(f(n,W)\) represents the maximum value that can be obtained by putting the first \(n\) items in a backpack with a capacity of \(W\). For each step of recursion, we have made a choice: select or not, which is equivalent to the case of each choice. Recursively traverse all nodes on the solution set tree, so that the correct solution of the 0-1 knapsack problem can be obtained.

However, there are actually a lot of repeated solutions, so we thought of setting up a two-dimensional array to save the results of each recursive solution, so that we don't need to repeat the solution next time. This is the memory recursion solution.

The solution of memory recursion is actually very close to dynamic programming. As mentioned earlier, the difference between the two is that one is top-down and the other is bottom-up. In fact, they can also be regarded as the relationship between recursion and loop. In theory, recursion and loop can be converted to each other. Therefore, by converting the recursive process into a loop by the memorized recursive recursion (dynamic transfer equation) and a two-dimensional array, the solution of the dynamic programming method can be obtained. For the solution of dynamic programming, the selection of each item can be regarded as a stage.

5. Conclusions
The key point of the dynamic programming method is to find the transfer equation, and this transfer equation is equivalent to the recursive formula in the recursive solution method. This article systematically introduces the basic ideas, problem-solving steps and application examples of the dynamic programming method, and specifically expounds the similarities and differences between the dynamic programming method and other algorithms and their conversion relations. Through the comparison with other algorithms, the essence of dynamic programming is analyzed and considered: use the solved problems to solve new problems.
References

[1] Anany Levitin. Design and analysis of algorithms reconsidered. 2000, 32(1): 16-20.
[2] Pferschy Ulrich, Scatamacchia Rosario. Improved dynamic programming and approximation results for the knapsack problem with setups. 2018, 25(2): 667-682.
[3] Dereventsov A. V, Temlyakov V. N. A unified way of analyzing some greedy algorithms. 2019, 227(12): 49-64.