CrossRef DOI of original article:

The Study and use of Dynamic Programming

Deepak Prajapat[1] and Aishwayra[2]

[1] Ajeenkya DY Patil University

Received: 1 January 1970 Accepted: 1 January 1970 Published: 1 January 1970

Abstract

When learning algorithms for the first time, dynamic programming is one area that is not well understood, but it is also a part that should be studied. It has been used effectively in numerous fields, such as controlling human movement, distributing hydroelectric resources, and gene sequencing. The dynamic programming principle is explained in detail in this article. Comparing it to other algorithms at the same time, we are able to comprehend dynamic programming's nature, as well as its benefits and drawbacks when compared to alternative techniques for problem-solving. On the basis of pertinent application examples, it then explores the dynamic programming problem-solving techniques and stages.

Index terms— knapsack problem, memory recursion, and dynamic programming.

1 I. Introduction

sing the dynamic programming technique, it is possible to solve the optimal solution problem for multi-stage decision-making. Not as "dynamic" as the name would imply. When attempting to resolve a practical issue, it establishes the starting point and breaks the large problem into smaller ones. The prior subproblem can be used to solve the current sub problem. The relationship between the present and previous subproblems, also known as the state transition equation, is the center of this problem and the source of its difficulty. After determining the equation for the state transition, the sub-solution problem is gradually between the bottom and the top of the problem's original state in order to resolve the larger overall issue.

2 II. Essential Concept of Dynamic Programming

Check to see if the situation at hand has ideal substructure features first, overlapping subproblem characteristics, and absence of consequences, this determines whether dynamic programming may be used to solve the issue. The term "optimal substructure" refers to the property that the best solution to a problem also incorporates the best solutions to all of its subproblems. When the problem is divided into subproblems, [1]; the overlapping sub-problem indicates that no aftereffect denotes that after the state of a specific stage is established, once some of the sub-problems formed each time are repeated, the subsequent decisions made by this state won't have an impact on it [2]. Dividing a task into several stages is the fundamental concept of using dynamic programming to solve problems, thus a stage can have more than one state. These states can be used to determine the outcome of this stage as well as the values of each state in the following stage. And so on, until the solution of the last stage is found, that is, the solution to the problem.

Generally speaking, when considering the issue, our approach should be top-down. We must resolve the issue with the earlier stage of the first issue in order to address the initial issue, while the earlier stage has a number of states. The selection of any one of these could be the answer to the initial issue. Which is what the transfer equation needs to assess, and these states are determined by the final step. repeat this process till the starting state. Yet the way the calculations are done is bottom-up. Beginning with the initial situation, Calculations are made to determine each state's solution during the first stage. Subsequently, using these conclusions, the states of the subsequent step can be determined until the conclusion of the previous stage's solution.

3 III. Connectedother Algorithms a) Greedy Method

Using greedy concepts is another highly effective approach to solving the optimal problem, in addition to dynamic programming. Therefore, in order to use a greedy method to address the problem, the issue must meet the criteria for greedy selection, that is, compared to the application of dynamic programming, local optimal selection [3] is more stringent., and can yield the overall best solution. The dynamic programming approach can typically solve the issues that can be resolved by the greedy technique, yet the greedy method might not be able to tackle all the issues that the dynamic programming method solves. It seems sensible to think about greed as a unique instance of dynamic programming. Greedy just consider the here and now, whereas dynamic programming also considers the past. The dynamic programming algorithm is essentially a variation of the divide and conquers strategy. They each break down a major problem into smaller ones and deal with each one separately. The dynamic programming method differs in that a subproblem may occur many more than once. Due to the fact that the sub-problems overlap, solving the latter problem also necessitates solving the first. Hence, we considered storing these subproblems so that we could easily access their solutions while tackling larger subproblems, eliminating redundant calculations to improve algorithm efficiency; the divide-and-conquer strategy works better with autonomous subproblems. When the subproblems are recursively solved one at a time and then combined to answer the main problem, However, The efficiency of the algorithms won't be as high as that of dynamic programming. approach. The issue with dynamic programming can also be solved with it. As a result, the divide and conquer strategy may be used to understand the dynamic programming approach.

4 Fig. 2: Example of Divide and Conquer c) Memory Recursion

Memory recursion is also utilized to tackle the problem with the concept of the space-for-time algorithm, similar to the dynamic programming approach, and they actually have the same essence. Yet, the dynamic programming approach works from the bottom up whereas the memory recursion solves problems from the top down. The two can typically be used interchangeably. The cache in memory recursion is analogous to the dp table in dynamic programming, thus in dynamic programming, the state transition equation is the same as a recursive calling. The conversion between the two is somewhat comparable to that between recursion and loop.

5 IV. Application a) Steps for Resolving Issues

Once an issue arises, the first thing we must consider is if dynamic programming can be used to address it. Considering if the answer is optimal is necessary if dynamic programming can be used to resolve the issue. Dynamic programming can be used to solve the problem if it meets the prerequisites of the ideal sub-structure, the similarity of sub-problem characteristics, and the absence of a consequence. We are now considering if it is best to employ dynamic programming to resolve this issue. Assume that there are n phases to this problem, and each stage contains m states. Recursion can be used to solve this problem This issue can be resolved using the greedy method when m is equal to 1if each stage's ideal state is derived from the optimal state of the stage before it; Dynamic programming can be used to address this problem if a state from a previous step serves as the foundation for the ideal condition at each level.

Once it has been determined that this problem can be solved using dynamic programming, it is broken down into many steps based on its specific characteristics. We must employ various states to reflect the problem's current objective reality once it reaches a given level of development. The transition equation, or link between a stage's current state and its predecessor stage's current state, is what we need to discover. Prior to that, we must first determine the beginning state and make sure the state we choose has no consequences. Find the best solution at each level in accordance with the transfer equation, and then locate the answer to the initial problem by finding the best solution at the last stage.

6 b) Application Examples

A well-known issue with dynamic programming is the 0-1 knapsack problem, which is also worthwhile understanding because it may be used to solve a variety of other problems. The issue is described in the following way: Given a rucksack with a capacity of W, n objects with weights w1, w2, and wn and values v1, v2, vn are present. Create a strategy for choosing a few of these goods to put in the rucksack. Either one of the items is chosen or not. The chosen goods must have the highest worth in addition to being able to fit in the rucksack. The first thought that comes to mind is typically a pretty violent recursive one. There are two options for each item: either place it in the backpack or do not place it in the backpack. This gives us the occurrence: f (n, W)=max (f (n-1,W), f (n-1, W-wn)+Vn). The largest value that is possible after packing the first n things into a bag with a W-liter capacity is represented by f(n, W) among them. We have made decisions for each recursive stage, including whether to select, which

7 V. Conclusions

Using the recursive solution approach, finding the transfer equation, which is the same as the recursive formula, is the main objective of the dynamic programming approach. This article presents the fundamental concepts, steps for solving problems, and examples of applications of the dynamic programming method, and specifically explains how the dynamic programming approach differs from other approaches in terms of conversion relations is discussed. The essence of dynamic programming is evaluated and taken into consideration through comparison with other algorithms: apply the solutions to old issues to solve new ones. The Study and use of Dynamic Programming means Citing the case of each decision. Recursively go through each node on the solution set tree to find the 0-1 knapsack problem's proper answer.

In reality, there are many repeated answers, therefore we came up with the idea In order to avoid having to repeat each recursive solution in the future, we created a two-dimensional array to store the results. The memory recursion approach looks like this.

Actually, dynamic programming and the memory recursion method are pretty similar. As was already mentioned, the two vary in that one is bottomup and the other is top-down. In fact, they may also be thought of as the way recursion and looping interact. Theoretically, loop and recursion are interchangeable. Consequently, the dynamic programming method's solution can be reached by turning with a twodimensional array and the dynamic transfer equation that was previously memorized, turning the recursive process into a loop. Each item's choice can be viewed as a stage in the dynamic programming problem.

8 Computer Algorithms, 2019, Computer Science

Press, Horowitz et al.

1

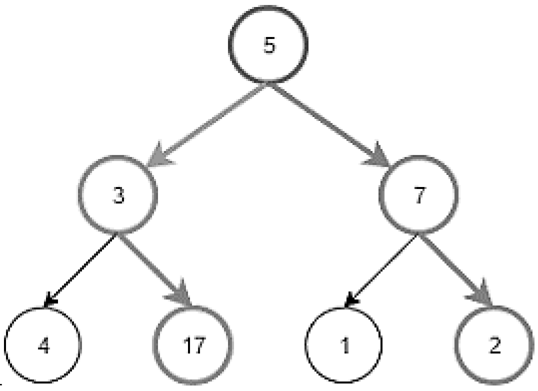

Figure 1: Fig. 1 :

[Polya ()] , *How to Solve It* G Polya (ed.) 2015. Princeton University Press.

[Algorithms and Sedgewick ()] , R Algorithms , Sedgewick . 2020. Addison-Wesley.

[Teaching Algorithms. SIGACT News (2015)] , *Teaching Algorithms. SIGACT News* December 2015. Baeza-Yates, R. 36 p. .

[Cormen ()] *Algorithms: An Introduction*, T Cormen . 1992.

[Brassard and Bratley ()] G Brassard , P Bratley . *Fundamentals of algorithms*, 2016. Prentice-Hall.

[Compared To What? : an Introduction to the Study of Algorithms J. Rawlins ()] 'Compared To What? : an Introduction to the Study of Algorithms'. *J. Rawlins* 2019. Comp. Sc. Press.

[R. Neapolitan and K. Naimipour. Jones and Bartlett (ed.) ()] *Foundations of Algorithms*, R. Neapolitan and K. Naimipour. Jones and Bartlett (ed.) 1997. (second edition)

[J. Bentley (ed.) ()] *Programming Pearls*, J. Bentley (ed.) 2016. Addison-Wesley.

[B. Kernigan and R. Pike (ed.) ()] *Programming Practice*, B. Kernigan and R. Pike (ed.) 2005. Addison-Wesley.

[Pferschy and Scatamacchia ()] *Results of improved dynamic programming and approximation for the setups knapsack problem*, Ulrich Pferschy , Rosario Scatamacchia . 2017. 25 p. .

[Rethinking algorithm design and analysis, Ananya Levitin ()] *Rethinking algorithm design and analysis, Ananya Levitin*, 2019. 32 p. .

[Should we teach the correct algorithm design techniques? Levitin, A. 179-183 in Proc. SIGCSE '99 (2013)] *Should we teach the correct algorithm design techniques? Levitin, A. 179-183 in Proc. SIGCSE '99*, March 2013.

[Dereventsov] *Temlyakov. a methodical approach to studying several greedy algorithms 2022*, D B Dereventsov , VF . 227 p. .

[The Algorithm Design Manual by Skiena, S. 1997] *The Algorithm Design Manual by Skiena, S. 1997*, Springer Verlag.

[The maximal rectangle problem Dr. Dobb's Journal (2021)] 'The maximal rectangle problem'. *Dr. Dobb's Journal* April 2021. Vandervooerde, D. 23 p. .