Advances in Social Science, Education and Humanities Research, volume 480
Proceedings of the 2020 5th International Conference on Modern Management and Education Technology (MMET 2020)

Analysis of Recursive Teaching of Computational Thinking

Ren Xiaokang, Yang Deyi*

School of I Northwest Normal University, Lanzhou, Gansu, China
**Corresponding author. Email: 237993036@qq.com*

ABSTRACT
The idea of recursion runs through the algorithm design itself and is a very important part of the program design. Learning algorithms is a good way to cultivate computing power. This paper analyzes computational thinking, the concept of recursion, algorithm thinking methods and steps based on recursive problems, and combines the procedural nature of algorithm design and computational thinking problem solving from five aspects, so as to analyze the thousands of factors between algorithm design and computational thinking. The inextricable connection enables students to strengthen their computational thinking skills while studying algorithm design courses; they can be more comfortable in solving programming problems.
Keywords: *computational thinking, recursion, divide and conquer, backtracking*

1. INTRODUCTION

Since Professor Zhou Yi-zhen, Carnegie Mellon University, put forward the concept of "computational thinking", the international and domestic education fields have set off an upsurge of research on computational thinking [1].Professor Zhou Yi zhen proposed: Computational thinking is a series of thinking activities covering the breadth of computer science, such as problem solving, system design, and understanding of human behaviour using the basic concepts of computer science. It is also foreseen that computational thinking will become an essential cognitive skill for everyone like reading, writing, and arithmetic. A report on computational thinking by the National Re-search Council (NRC) advanced a similar idea, that CT is cognitive skill which the "average person is expected to possess" [2]. Similarly, Bundy suggested that computational thinking concepts have been used in other disciplines via problem solving processes, and that the ability to think computationally is essential to every discipline [3]. With the Ministry of Education issued "Several Opinions on Further Strengthening the Undergraduate Teaching in Colleges and Universities", it is proposed that colleges and universities should actively promote research-based teaching and improve their innovative ability [4]. Nowadays, computational thinking has received widespread attention. As a good way to cultivate students' computational thinking, algorithm programming courses must have common characteristics with computational thinking. We recognize that the process of computational thinking is generally problem analysis, problem decomposition, and solution planning. The essence of computational thinking is abstraction and automation. Algorithm programming is a very critical course. Recursion is widely used in programming, and it is also a difficult point in many students' learning. Many students understand recursion, but they always encounter various problems in the process of programming, especially non-computers. Of students are very difficult to understand.

1.1. Basic understanding of recursion

Recursion is a process of directly or indirectly calling oneself. It is completed by two parts, one is the calling phase, and the other is the return phase. When the program is executed, the loop is gradually applied according to the setting of the conditions until the termination instruction is met. The process of returning to the previous instruction and finally getting the return value. There is an inseparable connection between recursion and loops. Recursion is often used to replace loops. Recursion has always been irreplaceable in program design. Combining computational thinking to perform recursive problem algorithm selection analysis requires a preliminary understanding of recursion, so what? Is it recursive? What is the idea of recursion? What are some examples of recursion in real life?

1.1.1. The concept of recursion

In fact, recursion is a kind of nested call function that you use to define yourself. In programming, you call your own process. If there is no termination condition, an infinite loop is formed, so recursion needs a conclusion. This conclusion is not to terminate the recursion, but to terminate the process of transmission and let the process of regression begin. When the substantive recursive function is called, the system allocates stack space for it. This process is the process of pushing variables onto the stack. At the end of each call, there will be a popping process, and the corresponding value at the top of the stack is assigned to the formal parameter. And variables, continue to start the next call return process until the end.

1.1.2. Recursive thinking

The idea of recursive problems is to decompose a large problem into smaller ones, and perform the same operations on smaller problems. Recursion has a very special problem among the small problems. This problem is the termination condition of recursion. The variable that controls the termination condition is called the recursive element. Usually, divide and conquer ideas and backtracking ideas are commonly used to solve recursive problems. If all the small problems decomposed by recursion are combined to obtain the final solution of the recursive problem, then this is a recursive problem solved by divide and conquer; if you decompose several Small problems, one loop and one loop, must be the next small problem that depends on the solution of the previous small problem, then this is a recursive problem solved by retrospective thinking. According to the description of the recursive thought, we can get the principle diagram of divide and conquer as shown in Figure 1, and the principle diagram of backtracking as shown in Figure 2.

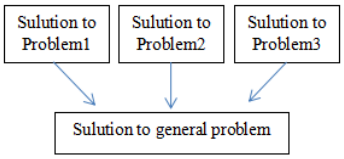

Figure 1 The principle diagram of divide and conquer

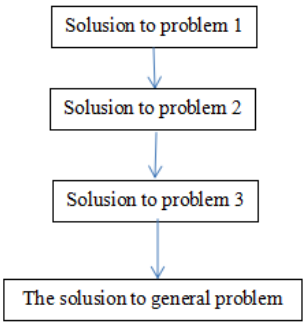

Figure 2 The principle diagram of backtracking

2. BACKGROUNDANALYZE THE RECURSIVE PROBLEM FROM THE PERSPECTIVE OF COMPUTATIONAL THINKING

In the learning process of students, they can understand the recursion problem by understanding the concept of recursion, and then analyze the recursive problem case, such as the most classic N factorial problem, the Tower of Hanoi problem, etc. After programming and executing the dynamic picture demonstration, it is completed Teachers often think that the problem is relatively simple from their own point of view, and students should understand more thoroughly after the dynamic demonstration. In fact, most students only remember the demonstration process. Better students understand the principle and have not reached a thorough understanding of how the problem occurs. Analyze the problem, solve the problem, and not when you encounter other similar problem-solving processes. Analyzed from the perspective of computational thinking, teachers need to downplay their own role, so that students can participate completely, through the substantial abstraction and automation of computational thinking to carry out problem analysis, problem decomposition, modeling, algorithm design, evaluation and reflection, from divide and conquer Look back at two angles to analyze the recursion problem.

2.1. The concept of computational thinking

Computational thinking was proposed by Professor Zhou Yizhen in 2006, who believed that computational thinking is a series of thinking activities using the basic concepts of computer science for problem solving, system design and human behavior understanding (wing, 2006). Since then, the American Association for Educational Technology and the Association of Computer Teachers have proposed professional vocabulary related to computational thinking, such as data collection, data representation, data analysis, problem decomposition,

abstraction, algorithms and programs, automation, simulation, and parallelization (ISTE&CSTA, 2011b). Professor Zhou Yizhen believes that computational thinking can be applied to various disciplines. It is a ubiquitous existence of human thinking like logical thinking. It will be called essential human skills like listening, speaking, reading and writing. Computational thinking will combine basic mathematical thinking in the process of application, and use computer science knowledge to complete the automatic solution of the problem [5].In recent years, the scholars believe that the computational thinking is one of the core qualities of information technology disciplines. Therefore, improving students' computational thinking ability is one of the core goals of computer teaching [6].

2.2. Basic analysis mode for recursive problems

There are two types of recursion: direct recursion and indirect recursion. Here we take the direct recursive printing of digital triangles as an example for analysis. This program realizes the printing of digital triangles by calling the function print. Each time its own function is called, the value of n is reduced by one from the previous call. When n is 0, it returns to printing.

2.3. Recursive problem analysis model combined with computational thinking

From the perspective of how students solve the problem combined with computational thinking, the problem model analysis of the printing digital triangle is mainly carried out from the following five steps: problem analysis, problem decomposition, abstract modeling, algorithm design, evaluation and reflection.

2.3.1. Problem analysis

First, analyze what kind of problem the problem is, whether it is a recursive problem, what is the scale of the problem, whether it can be executed recursively, and if the result required by the problem is executed recursively, given the input and output conditions, what is the termination condition.

2.3.2. Problem decomposition

The process of problem decomposition is actually relatively simple. It only converts large-scale problems into small-scale problems, and small-scale problems into smaller-scale problems, until it can't be transformed, but it is the same scale regardless of scale. The process is to reduce the scale until it can no longer be reduced.

2.3.3. Abstract Modeling

After decomposing the problem by scale, it is necessary to analyze the problems solved by the sub-problems of various scales. Each sub-scale problem is divided into two types. One is an equal relationship. The scales do not affect each other and are independent of each other. The solutions of each scale are combined to obtain The solution of the general problem, that is, this problem is the idea of divide and conquer, it is necessary to use the divide and conquer strategy to model the problem; the second is the child-parent relationship, with the result of the first question, there is an answer to the second question, that is, the question belongs to For backtracking thinking, it is necessary to use backtracking strategies to model the problem recursively.

2.3.4. Algorithm design and operation

According to the combination of problem analysis, problem decomposition, and abstract modeling, the algorithm design is carried out. First, the function framework is written, and then input and output conditions and termination conditions are added at the appropriate positions. The algorithm is realized through the automatic operation of the program. This process That is the process of automated execution.

2.3.5. Evaluation reflection

In the teaching process, a teaching program that feels good may not necessarily have a good teaching effect. The algorithm design and analysis process combined with computational thinking is even more essential to evaluate the teaching effect and evaluate the computational thinking method, and to carry out the teaching process through the teaching effect Reflection will enable students to improve their computational thinking ability and grasp the knowledge points.

3. CP ANALYSIS OF EXAMPLES OF ALGORITHMS COMBINED WITH COMPUTATIONAL THINKING

Taking computational thinking as the starting point, analyze the problem from five aspects.

3.1. Example analysis of the Tower of Hanoi problem

The Tower of Hanoi problem is a very typical example of recursion. Suppose there are three pillars, denoted by A, B, and C, respectively. It is required to move the plate on A to pillar C. At any time, the small

plate should be on the upper plate and the larger plate on the lower plate. Only one plate can be moved at a time. Partial processing of problems through computational thinking analysis models:

3.1.1 Problem analysis

The Tower of Hanoi problem is first analyzed briefly. Assuming that there are only four plates, the solution is obviously to first move the three plates on A to B using C as an intermediary, and then move the fourth plate on A to C, and finally A is an intermediary to move the three plates on B from B to C. We first reduce the size of the problem from N plates to 4 plates. The actual problem to be solved is the same because of the number of plates. We can't do it purely by calculation. We can only use recursive functions to make big problems smaller. Therefore, we are sure that this problem is a recursive problem and has a recursive nature. According to the conditions given in the title, the market is on top and one plate is moved at a time. The condition for recursive input, when will the recursion be terminated? The recursion ends when there is only one plate left.

3.1.2 Problem breakdown

The key to the decomposition of the Tower of Hanoi problem lies in the number of plates. According to the problem analysis, the first step is to move n-1 plates from A to C. The second step is to move the remaining plate to B. Finally Move N-1 plates through a to C, and only one plate can be moved at a time during the moving process, that is to say whether the plate is moved from A to B or the plate is moved from A to C during the moving process, Still move the plate from B to C. When the number of plates is reduced to 1, the smallest problem can no longer be resolved, that is, the plate moves at this time.

3.1.3 Abstract modeling

According to the problem analysis and problem decomposition, we can find that moving the plate is carried out in two situations, that is, when the number of plates is 1, it is moved, and when the number of plates is greater than 1, it cannot be moved, and the scale can only be reduced again to generate recursive functions. Move until the number of plates is gradually reduced to 1. Moreover, according to the model, it is divided into several problems that are smaller in scale, but belong to the same kind of independent problems. This problem is a recursive problem solved by the divide and conquer idea.

3.1.4 Algorithm design and operation

Through the above analysis and modeling, we can easily start programming based on problem analysis, problem decomposition, and abstract modeling. The function part of the execution code is as follows (c language programming):

```
void hanoi(int n,char A,char B,char C)
    {
        if(n==1)
         move(A,C);
        else
        {
                hanoi(n-1,A,C,B);
                move(A,C);
                hanoi(n-1,B,A,C);
        }
    }
```

3.1.5 Evaluation and reflection

The Tower of Hanoi problem is a typical divide-and-conquer problem. When we use computational thinking to solve the divide-and-conquer problem, is it feasible to also solve the problem of retrospective type? The answer is yes, when we try to solve the Fibonacci amount problem, it is still very appropriate.

4. CONCLUSION

Analyze the problem with the idea of computational thinking, start from the root of the problem, analyze, decompose, and model the problem, so that many students who are unable to start when they see programming can analyze the problem step by step according to different modules, and slowly Ideas, know where the key points are to solve the problem, after the model is established, you can write the code and run successfully. For students with strong programming ability, combining computational thinking to analyze recursive problems also enables them to have strong computational thinking solving skills when solving programming problems without panic, and more importantly, so that the teacher is no longer the role of a traditional teacher, The classroom is no longer a traditional classroom where teachers talk and listen to students. Through the integration of computational thinking, the role of teachers is downplayed, students' participation is strengthened, the quality of students' classrooms are improved, and their ability to solve problems after class is enhanced. The ability of computational thinking will gradually deepen into all disciplines and stages.

REFERENCES

[1] FAN Wen-xiang, ZHANG Yi-chun, LI Yi. A review of research and development of computational thinking at home and abroad. Journal of Distance Education. 2018, 2:3-15.

[2] Committee for the Workshops on Computational Thinking. Report of a Workshop on The Scope and Nature of Computational Thinking. The National Academies Press, 2010

[3] A. Bundy. Computational thinking is pervasive. Journal of Scientific and Practical Computing, 1:67–69, 2007.

[4] The Ministry of Education. "Some opinions on Further Strengthening Undergraduate Teaching in Colleges and Universities." China University Teaching, February, 2005. (In Chinese)

[5] Yong Yang, Chaoyan Zhu,and Huanda Lu."A Case Study of C Language Programming Teaching based on Computational Thinking". Proceedings of 1st International Conference on Education and Educational Development (EED 2020). Ed. BCP, 2020, 139-142.

[6] Jing G, Chang-Lin H , Xin-Ting Z . Study on the Computational Thinking Ability Development Model and the Teaching Program[J]. modern educational technology, 2018.