Toward the use of Generative AI to develop Computational Thinking by supporting Problem Decomposition

Davide Ponzini[1,2,*], Giovanni Adorni[1], Giorgio Delzanno[1] and Giovanna Guerrini[1]

[1]*University of Genoa, Dipartimento di Informatica, Bioingegneria, Robotica e Ingeneria dei Sistemi, via Dodecaneso 35, Genoa, 16145, Italy*
[2]*University of Genoa, Dipartimento di Lingue e Culture Moderne, piazza Santa Sabina 2, Genoa, 16124, Italy*

Abstract
We present a possible applications of generative AI to support a Computational Thinking approach to learn programming principles with a particular focus on problem decomposition. Our approach is based on a visual tool that guides students in decomposing a problem in smaller task, prompting ChatGPT on demand via predefined queries designed via a preliminary prompt engineering experimental phase. The tool also provides the possibility of prompting ChatGPT to generate code in a bottom-up manner, reusing functions generated in previous steps. We illustrate here the main ideas with the help of a case-study.

Keywords
Generative AI, ChatGPT, Computational Thinking, Problem Decomposition

1. Introduction

The role of Generative AI in computing education is one of the mostly debated issues in the last year, as evidenced by works such as [1]. Although the advantages and disadvantages of using tools such as ChatGPT[1] to support the teaching of computer-related subjects are not fully clear yet, there is a growing consensus that the technological progress of Generative AI will require an adaptation of the educational methods currently employed.

In the context of introductory and advanced programming courses, tools such as ChatGPT, which are already integrated in the most common software development tools (take as an example GitHub's Copilot) [2] are frequently exploited. These tools are designed in order to help the users to best formulate the questions submitted to the prompt.

According to the current literature on computing education, such as [1, 3, 4, 5, 6, 7], the most common use of Generative AI is to generate code or to explain specific features of existing or generated code. The use of Generative AI to generate solutions to assignments and exercises is indeed a critical issue to be taken into consideration to avoid plagiarism and negative effects on student learning outcomes.

In our research, we are interested in possible applications of Generative AI to develop a Computational Thinking approach in learning to program.

Computational Thinking [8] can be defined as

> *"The thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent."* [9]

Wing emphasized that Computational Thinking is a fundamental component of Computer Science, where its abstractions can be executed by computational agents. The shift from solving problems to effectively solving problems and making something perform computations for us distinguishes Computational Thinking from other disciplines.

One of the characteristics that defines Computational Thinking is problem decomposition, i.e., the capability of solving a complex problem by breaking it down into progressively smaller tasks.

In this paper we focus our attention on the possible use of Generative AI to develop Computational Thinking skills by supporting the process of Problem Decomposition [10] with the help of visual tools.

In particular, we examine the potential of ChatGPT to assist students in analyzing and decomposing problems into smaller subtasks, employing a top-down approach. This is followed by the generation of code for a possible implementation, using a bottom-up approach and potentially reusing the code generated in previous steps. Our approach is based a visual tool that presents the current decomposition in form of a labeled tree, in which nodes contain names and descriptions of each subtask. Starting from the root node with the description of the

Ital-IA 2024: 4th National Conference on Artificial Intelligence, organized by CINI, May 29-30, 2024, Naples, Italy
*Corresponding author.
davide.ponzini@edu.unige.it (D. Ponzini); giovanni.adorni@unige.it (G. Adorni); giorgio.delzanno@unige.it (G. Delzanno); giovanna.guerrini@unige.it (G. Guerrini)
0009-0006-4282-9652 (D. Ponzini); 0000-0003-3933-4377 (G. Adorni); 0000-0001-7030-1050 (G. Delzanno); 0000-0001-9125-9867 (G. Guerrini)

[1]https://chat.openai.com/

problem to solve, the user chooses which nodes to further decompose. The decomposition process is designed to be interactive and fully guided by the user until the solution is decomposed in a collection of subtasks that the user is capable to implement. The decomposition of a given task, i.e., a given node of the tree, can be provided manually by expanding the corresponding subtree with a set of labeled nodes inserted by the user. In addition it can also be generated via a predefined query to ChatGPT. The query formulation is based on a preliminary prompt engineering experimental design phase that seems to provide good results in the most common coding tasks taken from the literature on introductory programming courses. In our approach, the prompt engineering task is entirely hidden to the user and ChatGPT is considered as an oracle to support students during the decomposition process.

If needed, in a second phase, users can prompt ChatGPT to generate the code of the solution in a bottom up manner. The predefined query generated by our tool ensures that functions generated for a given subtask can be reused for generating the code of the current task.

The implementation of the tool allowed us to evaluate how well ChatGPT is able to decompose a high-level task into smaller, easier-to-solve subtasks, and to implement decomposed tasks using a bottom-up approach.

In the paper we present the methodology, the proposed tasks, and the preliminary results with the help of an example.

The paper is structured as follows: In Section 2, we outline the background of our study, focusing on the effects ChatGPT has on education and computational thinking. Section 3 describes the proposed methodology and experiment setup. Section 4 reports, focusing on a case study, the results of our first experiments and discusses their limitations. Finally, in Section 5 we address conclusions and future research directions.

2. Background

ChatGPT has shown remarkable capabilities in solving programming tasks across a range of different programming languages, particularly when the instructions are presented in a clear and unambiguous manner [7, 6]. However, it is less effective when confronted with more complex requests, particularly when the questions are not structured in an optimal manner or when the information provided in the prompt is limited [6].

Even though the tool shows great potential in the educational domain, it is used by a limited number of students for educational purposes [11]. One of the reasons is that education regarding LLMs usage is very limited and students often don't know how to properly use these tools to support learning [11, 5]. Furthermore, a number of students may prefer not to use the tool, believing it will lead to reduced learning [11].

Among the CS students who employ the tool, the most common usage is for generating code, followed by debugging. Explaining difficult concepts falls in third place [12]. Students who use ChatGPT when facing programming assignments tend to show higher efficacy and computational thinking skills [13].

Currently, ChatGPT is mainly employed to automatically provide students with feedback when programming a task. Feedback can be provided in many forms, such as coding hints for the next instruction [4] and code or error explanations [14].

When confronted with more complex problems, ChatGPT is less adept at providing reliable solutions. As a result, the user is often required to manually break down the problem into smaller components and subsequently assemble the solution [15].

3. Methods

To evaluate the support offered by ChatGPT to decompose a high-level task into smaller, easier-to-solve subtasks, we designed a set of prompts for ChatGPT, as well as a custom visualization tool to display the results and allow the user to interact with the tool.

3.1. ChatGPT Prompts

Decomposing a task into subtasks To decompose a broad task into more refined subtasks we experimented with several prompting styles.

Our initial approach was to include the current task decomposition in the prompt, so that ChatGPT would be aware of which tasks had already been decomposed and how they were decomposed at each iteration. However, this approach proved ineffective as ChatGPT struggled to understand the decomposition. Various notations were attempted, including JSON and ad hoc syntax, similar to regular tree expressions, to describe the current structure of the decomposition, but none was found to have a positive impact. Ultimately, two distinct prompts were utilized, and ChatGPT's conversational memory was relied upon to maintain the current decomposition state.

The initial prompt is used when a user first wishes to decompose a problem. It includes a user-provided problem description and detailed instructions for ChatGPT on how to accurately decompose it. The main requirements we identified are:

- Specifying that the task needs to be decomposed into a small number of subtasks. This requirement is crucial as ChatGPT has a tendency to immediately decompose the problem into approximately ten steps, with emphasis on the process rather than on the reasoning behind the task.

- Requiring the fields "name" and "description" to be similar to the ones initially provided. This is important for keeping stylistic consistency between all tasks.
- Returning the results in JSON format. This is important for using the output in our visualization tool. We also noticed that this requirement made the title and description more precise and each subtask being assigned a unique name.
- Requiring that subtasks of a given task did not contain any element of other tasks. This is needed to avoid task mix-ups.
- Ensuring there are no missing steps in the decomposition, i.e., solving all the subtasks is equivalent to solving the original problem.

We also included an instruction indicating not to decompose a task in case no reasonable decomposition can be made (for example for basic tasks). However, this proved ineffective, as ChatGPT always ended up decomposing even the simplest tasks.

A simpler prompt has been designed for decomposing subtasks. The prompt references the task by name and requests that the same process be repeated for that task. It is important to note that each task decomposed using this prompt has been originally generated by ChatGPT.

Implementing a task Once the user is satisfied with the current decomposition, they can start implementing the subtasks following a bottom-up approach. The prompt requires ChatGPT to implement the given task in a specific programming language, which can be selected by the user.

We identified two requirements, which are:

- Using, whenever possible, functions already generated by ChatGPT for other tasks. This requirement is crucial, since it can highlight the interaction between a task and its subtasks.
- Not writing the implementation of functions which have already been implemented, to avoid redundant code.

3.2. Visualization tool

The tool we have created enables users to easily visualize tasks and explore their decomposition into subtasks. Users have the option to display or hide subtasks for a given task and, if available, review its properties and implementation. Alternatively, ChatGPT can be utilized by the tool to automatically decompose or implement any task. Task decomposition is only performed upon user request, as each user's knowledge base varies and some tasks may be clear to some users but not to others.

Tasks can also be marked as "solved", indicating that the user has fully understood the task and no further decomposition is needed. For the sake of usability, solved tasks are presented in a distinct color, allowing for a clear representation of the current understanding of the problem at all times.

Tasks can also be edited, created, or deleted manually. However, ChatGPT does not currently reflect these actions. Automatic decomposition or implementation of these tasks is not currently available.

4. Case Study

The tool was employed to decompose and implement a small number of tasks. This section presents the results for the task *"Write a Python program to find the most trending videos, given a CSV file containing each visualization"*. The task has been intentionally formulated as a broad request, to test if the program would be able to correctly identify its subtasks.

4.1. Task decomposition

Figure 1 shows the decomposition of the task. ChatGPT decomposed the main task in the following subtasks, which is similar to how we would have manually decomposed it.

- **Read CSV file:** *Write a Python function to read the CSV file containing video views data.*
- **Parse CSV file:** *Write a Python function to parse the data from the CSV file and extract relevant information such as video IDs and view counts.*
- **Calculate trending score:** *Write a Python function to calculate a trending score for each video based on its view count and possibly other factors such as upload date.*
- **Sort videos by trending score:** *Write a Python function to sort the videos based on their calculated trending scores in descending order.*
- **Retrieve top trending videos:** *Write a Python function to retrieve the top N videos with the highest trending scores, where N is a parameter.*

We chose not to decompose basic tasks, such as reading, parsing, sorting, or displaying data, as they can be

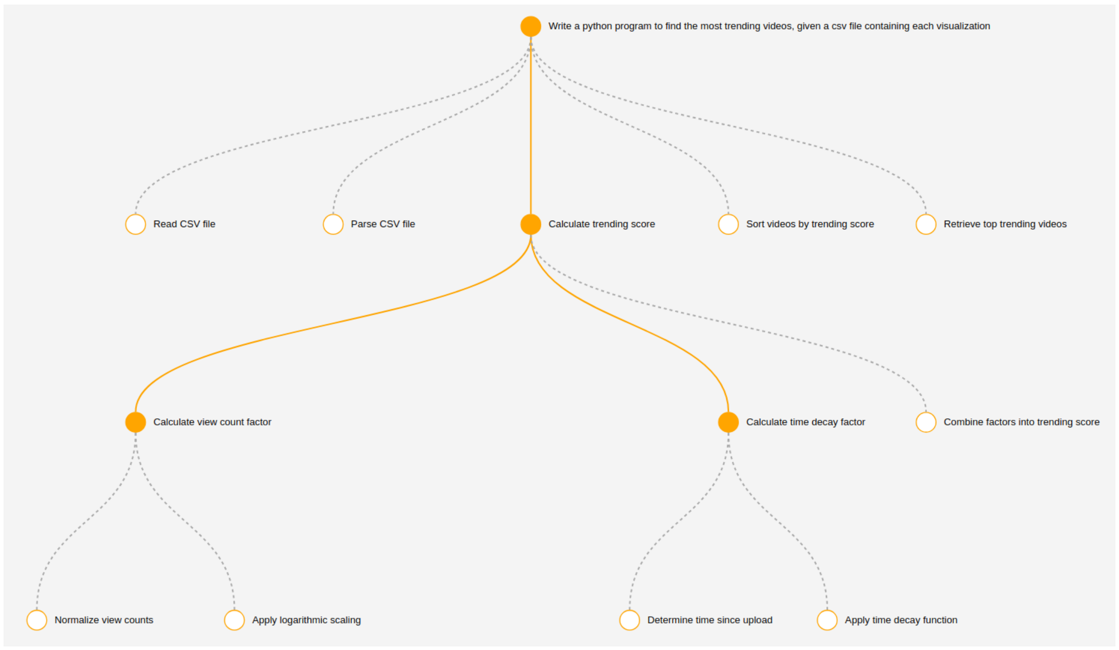

Write a python program to find the most trending videos, given a csv file containing each visualization
Read CSV file
Parse CSV file
Calculate trending score
Sort videos by trending score
Retrieve top trending videos
Calculate view count factor
Calculate time decay factor
Combine factors into trending score
Normalize view counts
Apply logarithmic scaling
Determine time since upload
Apply time decay function

Figure 1: Decomposition for the problem *"Write a Python program to find the most trending videos, given a CSV file containing each visualization"*.

easily accomplished using existing functions in Python. Instead, we decided to further decompose the task "Calculate trending score" because its implementation was not easily understandable. ChatGPT decomposed the task into:

- **Calculate view count factor:** *Write a Python function to calculate a factor based on the view count of a video, possibly using logarithmic scaling to give more weight to videos with higher view counts.*,
- **Calculate time decay factor:** *Write a Python function to calculate a factor to represent the time decay of a video's popularity, considering how recently it was uploaded.*
- **Combine factors into trending score:** *Write a Python function to combine the factors calculated in the previous steps into an overall trending score for each video.*

which is consistent with how we would have simplified the task. Further decompositions of its subtasks, as shown in Figure 1, were also consistent with our expectations.

4.2. Task implementation

After obtaining the decomposition shown in Figure 1, we instructed ChatGPT to implement the tasks using a bottom-up approach. This involved starting from the smallest subtasks and subsequently progressing towards the top-level task.

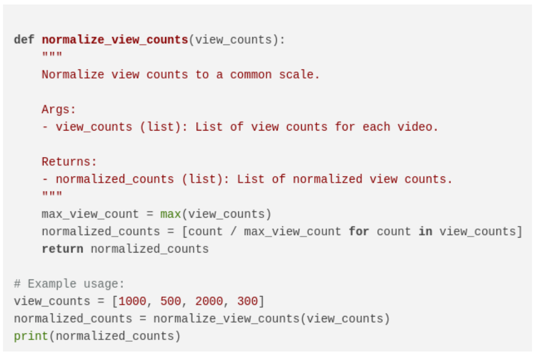
For each subtask, ChatGPT created a function which solved the given problem. Each implementation also contained a docstring describing its purpose, parameters and returned value, as well as a usage example, as shown in Figure 2.

For tasks that were previously decomposed, ChatGPT was able to accurately recall and utilize the functions it had previously implemented, as shown in Figure 2b.

4.3. Limitations

The main limitation of this study is the relatively small number of problems that were decomposed and implemented using our system. It is possible that some tasks may present unforeseen challenges that require further tuning of our prompts or different approaches.

A further limitation of the study is the small number of users who have tested the system. By testing the system with a larger number of users, it would be possible to ascertain whether certain tasks present issues when decomposed.



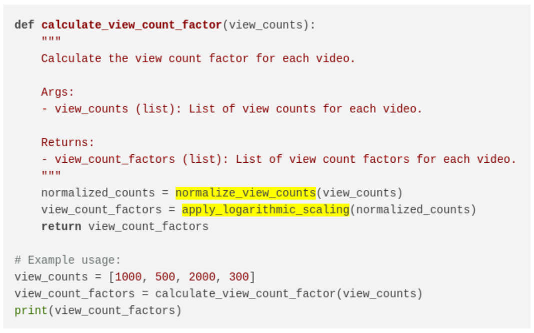
```python
def normalize_view_counts(view_counts):
    """
    Normalize view counts to a common scale.

    Args:
    - view_counts (list): List of view counts for each video.

    Returns:
    - normalized_counts (list): List of normalized view counts.
    """
    max_view_count = max(view_counts)
    normalized_counts = [count / max_view_count for count in view_counts]
    return normalized_counts

# Example usage:
view_counts = [1000, 500, 2000, 300]
normalized_counts = normalize_view_counts(view_counts)
print(normalized_counts)
```

(a) Implementation for task *“Normalize view counts”*.



```python
def calculate_view_count_factor(view_counts):
    """
    Calculate the view count factor for each video.

    Args:
    - view_counts (list): List of view counts for each video.

    Returns:
    - view_count_factors (list): List of view count factors for each video.
    """
    normalized_counts = normalize_view_counts(view_counts)
    view_count_factors = apply_logarithmic_scaling(normalized_counts)
    return view_count_factors

# Example usage:
view_counts = [1000, 500, 2000, 300]
view_count_factors = calculate_view_count_factor(view_counts)
print(view_count_factors)
```

(b) Implementation for task *“Calculate view count factor”*. The functions highlighted in yellow have been created when implementing its subtasks.

Figure 2: Results of task implementation.

Additionally, as ChatGPT is frequently updated, different prompts may be required to obtain the same output.

5. Conclusions & Future Work

The developed tool demonstrates potential for decomposing tasks into smaller subtasks and implementing them using a bottom-up approach. The preliminary study decomposed a limited number of problems, but the results seem quite promising. ChatGPT correctly identified how to decompose the main problem into smaller subtasks and provided a well-documented implementation of each step.

Future directions for our work include:

- Integrating the tool with ChatGPT’s APIs to fully automate decomposition and implementation functionalities;
- Enabling users to fully modify existing decompositions and implementations;
- Testing our system with a larger number of problems;
- Defining precise evaluation criteria for proper task decomposition;
- Measuring if the tool can improve computational thinking learning skills in users.
- Using the tool to identify weaknesses and misconception of students when facing a sequence of tasks that require common solving techniques.

References

[1] P. Denny, J. Prather, B. A. Becker, J. Finnie-Ansley, A. Hellas, J. Leinonen, A. Luxton-Reilly, B. N. Reeves, E. A. Santos, S. Sarsa, Computing education in the era of generative AI, Commun. ACM 67 (2024) 56–67. URL: https://doi.org/10.1145/3624720. doi:10.1145/3624720.

[2] C. Ebert, P. Louridas, Generative AI for software practitioners, IEEE Software 40 (2023) 30–38.

[3] C. Zastudil, M. Rogalska, C. Kapp, J. Vaughn, S. MacNeil, Generative AI in computing education: Perspectives of students and instructors, in: IEEE Frontiers in Education Conference, FIE 2023, College Station, TX, USA, October 18-21, 2023, IEEE, 2023, pp. 1–9. URL: https://doi.org/10.1109/FIE58773.2023.10343467. doi:10.1109/FIE58773.2023.10343467.

[4] L. Roest, H. Keuning, J. Jeuring, Next-Step Hint Generation for Introductory Programming Using Large Language Models, in: Proceedings of the 26th Australasian Computing Education Conference, ACE ’24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 144–153. doi:10.1145/3636243.3636259.

[5] B. Qureshi, Exploring the Use of ChatGPT as a Tool for Learning and Assessment in Undergraduate Computer Science Curriculum: Opportunities and Challenges, in: 2023 9th International Conference on e-Society, e-Learning and e-Technologies, 2023, pp. 7–13. doi:10.1145/3613944.3613946, arXiv:2304.11214 [cs].

[6] E. L. Ouh, B. K. S. Gan, K. Jin Shim, S. Wlodkowski, ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness

in an undergraduate Java Programming Course., in: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, ACM, Turku Finland, 2023, pp. 54–60. doi:10.1145/3587102.3588794.

[7] C. Geng, Y. Zhang, B. Pientka, X. Si, Can ChatGPT Pass An Introductory Level Functional Language Programming Course?, 2023. doi:10.48550/ARXIV.2305.02230.

[8] J. M. Wing, Computational thinking, Commun. ACM 49 (2006) 33–35. URL: https://doi.org/10.1145/1118178.1118215. doi:10.1145/1118178.1118215.

[9] J. M. Wing, Computational thinking and thinking about computing, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366 (2008) 3717–3725.

[10] P. J. Rich, G. Egan, J. Ellsworth, A framework for decomposition in computational thinking, in: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 416–421. URL: https://doi.org/10.1145/3304221.3319793. doi:10.1145/3304221.3319793.

[11] H. Singh, M.-H. Tayarani-Najaran, M. Yaqoob, Exploring Computer Science Students' Perception of ChatGPT in Higher Education: A Descriptive and Correlation Study, Education Sciences 13 (2023) 924. doi:10.3390/educsci13090924.

[12] A. Ouaazki, K. Bergram, A. Holzer, Leveraging ChatGPT to Enhance Computational Thinking Learning Experiences, in: 2023 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), 2023, pp. 1–7. doi:10.1109/TALE56641.2023.10398358.

[13] R. Yilmaz, F. G. Karaoglan Yilmaz, The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation, Computers and Education: Artificial Intelligence 4 (2023) 100147. doi:10.1016/j.caeai.2023.100147.

[14] K. Kuramitsu, Y. Obara, M. Sato, M. Obara, KOGI: A Seamless Integration of ChatGPT into Jupyter Environments for Programming Education, in: Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E, ACM, Cascais Portugal, 2023, pp. 50–59. doi:10.1145/3622780.3623648.

[15] A. Sane, M. Albuquerque, M. Gupta, J. Valadi, ChatGPT Didn't Take Me Very Far, Did It?, in: Proceedings of the ACM Conference on Global Computing Education Vol 2, CompEd 2023, Association for Computing Machinery, New York, NY, USA, 2023, p. 204. doi:10.1145/3617650.3624947.