Students’ Thinking of Recursion: When Do They Use It And Why?

Adi Levy Conlogue

RISE Center, University of Maine, Orono, Maine

Introduction

Recursion is a fundamental key concept in Computer Science. It is common to present recursion during the first programming course as well as to focus on traditional, functional examples such as calculating factorials or Fibonacci series. It has been recommended that students learn and apply recursion as it is recognized as an important topic to learn and study (ACM, 2001). Some tasks may be properly solved using recursion while others can be solved using iteration. Every recursive solution could be implemented using iteration and vice versa; however some iterative solutions can be complicated whereas the recursive solution is simple.

Research has documented that recursion is difficult to learn, comprehend and apply in problem solving activities (e.g., Gotschi, Sanders & Galpin, 2003; Haberman & Averbuch, 2002). It is also hard to teach (e.g., Bruce, Danyluk & Martag, 2005; Velozquez-Irazuber, 2000). Teaching recursion can be challenging largely because students have difficulties envisioning the abstract concept (Duyn et al., 2000). Research has shown that students adhere to iteration even though a recursive solution could be more efficient, clean, and more appropriate. Little is known about students’ understanding and thinking about recursion.

The goal of this research is to examine when college students choose to use recursion as a strategy to solve algorithmic tasks while programming, and why they choose to use it.

Research Questions

- Do students choose to solve algorithmic tasks using recursion or iteration and what are the reasons for their choice?
- What reasons do students give for when they refrain from using recursion as a method to solve an algorithmic task?

Research Design and Methods

Clinical interviews (Hunting, 1997) were conducted with 17 participants who were undergraduate and graduate students from two majors (Computer Science and Computer Engineering). Participants solved a variety of tasks, using their preferred programming language or pseudo code. The interviews were audio recorded and transcribed. The data were analyzed using Grounded Theory (Glaser & Strauss, 1967).

Data Analysis & Results

At a later point during the interview the participant was asked to solve the same question using the method he/she did not use during their first response.

Proper solutions are:

<table>
<thead>
<tr>
<th>Iterative Solution</th>
<th>Recursive Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (int n, int x) int Result = x; int i = 1; For i to n do Result = Result x i; Return Result;</td>
<td>Power (int n, int x) If (n=0) then Return 1; Else Return (n- Power(n-1, x));</td>
</tr>
</tbody>
</table>

The first step of the analysis was to group the responses according to the method of solution. The data were categorized as either an iterative solution or as a recursive solution. The second step was to check for correctness of the solution. The same analysis was performed, when the participants solved the task using the other method.

The figure in the next columns shows the change in the participants responses from the 1st attempt to the 2nd attempt.

As the data were analyzed, categories were formed to try to shed light on why the participants choose to solve the task in an iterative approach. The following categories were developed:

- Class-related reasons.
- Recursion can cause stack overflow.
- More confident with iteration / not comfortable with recursion.
- First introduced to iteration / most practiced.

Research features - Hard to visualize

<table>
<thead>
<tr>
<th>Categories</th>
<th>%</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class-related reasons</td>
<td>11%</td>
<td>“I really do think it’s because I’ve been doing a ton of dynamic programming lately. I’m modeling everything recursively and then just... iterates through to solve the problem.”</td>
</tr>
<tr>
<td>Recursion can cause stack overflow</td>
<td>29%</td>
<td>“Because you are not relying on the call stack for this... You know, for large trees you can overflow the call stack.”</td>
</tr>
<tr>
<td>More confident with iteration / Not comfortable with recursion</td>
<td>29%</td>
<td>“That’s kind of like the first thing I just go to, why I think it is what I am more comfortable with.”</td>
</tr>
<tr>
<td>First introduced to iteration / Most practiced</td>
<td>29%</td>
<td>“Probably because that is what I use first introduced in... in programming books.”</td>
</tr>
<tr>
<td>Recursion features</td>
<td>29%</td>
<td>“It makes a whole lot more... it’s a lot harder to visualize I guess because you are going down through this recursion tree.”</td>
</tr>
</tbody>
</table>

Conclusions

This study shows that students choose to approach algorithmic tasks in an iterative way, rather then solve them recursively.

This study follows previous researchers findings that students adhere to iterative patterns due to their confidence with iteration.

Results indicate additional reasons that students might refrain from solving a problem in a recursive approach due to the following:

- If using recursion there is the possibility of error due to stack overflows.
- Recursion is a very abstract concept and it might be hard to track and visualize.
- Students might be approaching an algorithmic problem with a certain method depending on the class they are enrolled in.
- Students believe they are more prone to use an iterative approach because it is the first method they learned in classes and in programming books.

Since these findings presented here are part of a larger study, further analysis will be conducted in order to support claims made in this poster and verify that no additional reasons exist.

Literature Cited


