2024-06-07 / Reading Comprehension / English for Software Development

Chapter 13. A Matter of Interpretation

[Rewriting History; first to third paragraphs]

An algorithm as a tool for solving problem is of interest only if it can solve a number of related problems. If a specific algorithm could solve only one problem, such as finding the shortest route from your home to work, you could execute the algorithm once and then remember the route and forget the algorithm. If, [ ], the algorithm is parameterized and can find shortest routes between different places of interest, it becomes very useful, since it is [ ] in many situations.

When an algorithm is executed, the resulting computation works on input values that are [ ] [ ] the parameters. The getting-up algorithm in Chapter 2 consists of the instructions “Wake up at *wake-up-time*.” To execute that algorithm, a concrete time value such as 6:30 AM must be applied (for example, by setting the alarm), and so the instruction becomes “Wake up at 6:30 AM”, obtained by substituting the value 6:30 AM [ ] the parameter *wake-up-time* in the algorithm.

The substitution mechanism applies to all algorithms and their parameters: [ ] cups of water for making coffee, pebbles for finding a path, weather conditions for weather reports, and so on. Of course, parameter substitution also applies to recursive algorithms. For example, quicksort and mergesort require the list that is to be sorted as input; binary search has two parameters, the item to [ ] and the data structure (tree or array) to [ ] the search in; and the algorithm *Count* takes the list to be counted as input for its parameter.

[A Smaller Footprint; first and last paragraph]

Substitution is a simple mechanism for producing a trace of a computation, which is basically a sequence of snapshots of intermediate results or states. It works for nonrecursive and recursive algorithms [ ], but it is particularly useful for recursive algorithms because it eliminates self-reference and systematically turns a descriptive recursion [ ] a corresponding unfolded one. When we are only interested in the result of a computation and not in the intermediate steps, substitution is doing more than is needed, but the value of a substitution trace [ ] its ability to provide an explanation of the computation that has taken place. […]

Substitution and interpretation are two methods to understand the execution of algorithms, in particular, of recursive algorithms. Substitution is simpler, since it only works with a trace that is rewritten step-by-step, [ ] interpretation employs an auxiliary stack. Substitution mixes code and data, whereas interpretation keeps them cleanly separated, which simplifies the extraction of simple traces. In the case of unbounded recursion, substitution produces [ ] useful, whereas interpretation does not terminate.

[Dopplegängers Get More Done; last paragraph]

It's actually fun to execute quicksort and mergesort with the help of a group of people. To execute quicksort, all people line up in a queue, and the first person starts sorting by applying the second equation and splitting the list of all elements [ ] two sublists depending on whether they are smaller than the first element. She keeps the first element and [ ] each sublist to a new person from the

queue who performs quicksort on the list handed to him and possibly also recruits new people from the queue for the sorting of sublists. A person who is handed an empty list is [ ] done and can return that empty list, following the definition of the first equation. Once a person has finished sorting her list, she hands the sorted list back to the person who gave her the list. And everyone who receives a sorted sublist as a result creates his own sorted list by placing the list with the smaller elements before x and the list with the larger elements after x. If we stop the recursion when lists have only one element, then we need as many people to sort the list as the list contains elements, because each person hands on to only one element. This strategy may seem a [ ] of resources for simply sorting a list, but with ever-decreasing computing costs and increasing computing power, it illustrates the power of divide-and-conquer and shows that many hands make light work.