1.
MATCH THOSE SOCKS

Not for Distribution

Baroness Margie Wana is a member of a formerly influential Viennese family who was recently indicted for smuggling Kinder Surprise eggs into the United States. She now works as an au pair in Bern and is folding clothes for the first time. Margie is shocked to discover that each member of her host family sweats through a pair of socks every half hour, making finding and matching pairs more time-consuming than she ever imagined. On the plus side, they have different shoe sizes and like different colors.

Hint: While there might be several tasks here, perhaps start with the fundamental one.

Have you ever thought about how essential a biological feature *memory* is in humans? The image of someone leaning back into her chair, putting one hand to her forehead, and pressing her eyes shut as she calls to mind a verse or an equation or a telephone number—that's the quintessential human. Imagine the struggle of having to go through life without that feature, as do sufferers of dementia. For starters, you

9780735222120_BadChoices_TX.indd 7 2/13/17 11:59 AM

would end up having to repeat a lot of the same work. Like in *Memento*, where every morning the protagonist has to fill up his mind all over again with all the bits of information that he needs to carry out his primary task.

I mention this at the outset because of the fact that faster methods of solving problems are often faster because they happen to leverage memory.* Consider AlphaGo, which last year beat a champion at the game of Go thanks to its ability to learn not only from expert humans but also from itself, thus amassing a greater memory from which to work.† Put differently, many of the faster ways of solving problems that we will encounter in this book, simple though they are, are fast because of their ability to avoid doing the same action on the same thing multiple times.

Let's not get ahead of ourselves though. Back to the socks, and to poor old Margie Wana, the irony of whose name was lost on the federal agents who recently seized her stash of chocolate goodness. Margie is facing the daunting task of matching pairs of socks from a humongous pile of clothes. Let us focus on one of several tasks that exist here and consider two possible methods for taking on that task:

Not for Distribution

* A fact that is sometimes characterized with the phrase, "trading memory for time."

† This approach was pioneered at the University of Toronto a decade ago and is referred to as deep learning.

9780735222120_BadChoices_TX.indd 8 2/13/17 11:59 AM

OBJECTIVE: MATCH THE PAIRS OF SOCKS IN THIS PILE OF CLOTHES.

METHOD 1: PICK A SOCK, LOOK FOR ITS MATCH IN THE PILE, PUT IT TO ONE SIDE. THEN PICK ANOTHER SOCK, LOOK FOR ITS MATCH IN THE PILE, AND PUT IT TO ONE SIDE. AND SO ON.

METHOD 2: PICK A SOCK, PUT IT TO ONE SIDE. PICK ANOTHER SOCK. IF IT MATCHES ANY OF THE ONES SHE HAS PUT TO ONE SIDE, MATCH IT. OTHERWISE, ADD IT TO THE LINE OF UNMATCHED SOCKS, LUMPING IT WITH SOCKS THAT HAVE THE SAME COLOR OR SIZE.*

Not for Distribution

* Note that with both methods, we ignore the task of separating socks from non-socks because we're focused on the fundamental task of matching socks.

9780735222120_BadChoices_TX.indd 9 2/13/17 11:59 AM

Before reading any further, I would suggest working through these scenes using pen and paper, props, or whatever else you feel comfortable with. Think about what achieving the objective entails in terms of individual steps and assumptions. Try to do that for all the scenes that follow.

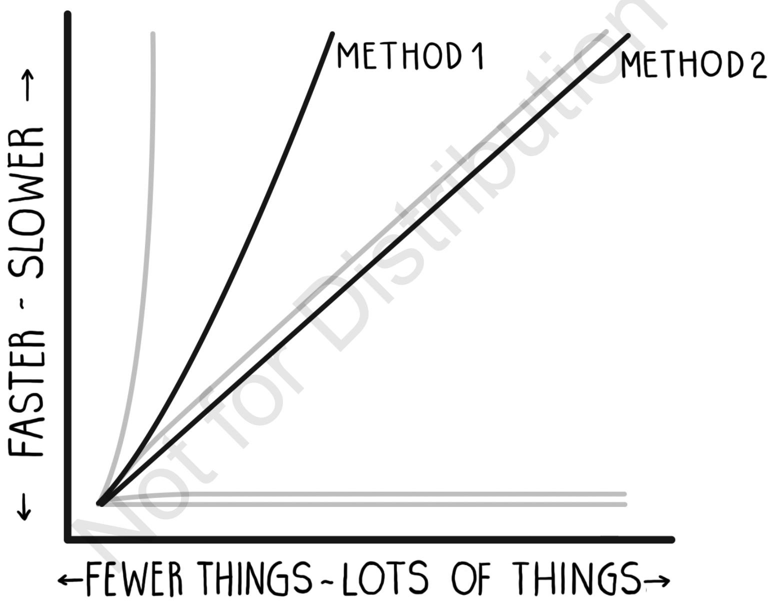
With a pile of, say, four pairs of socks, it doesn't really matter which method Margie uses—she will be done fairly quickly. Now imagine Margie with hundreds of socks in front of her. If she opts for the first method, the chance that she will come across the same old sock over and over again is quite high, since she never takes it out of the pile. When she first comes across it, she simply doesn't glean any information from it. With the second method, however, she keeps a line of unmatched socks to one side, thus ensuring that she only ever comes across a sock in the pile once. The second method, therefore, ends up being faster because of its reliance on memory. More precisely, because of what's sometimes called a *lookup table* or *cache*. Though it need not be, it is useful to think of a lookup table as a collection of unique identifiers (keys) each pointing to some associated item of data (values) where you, quite literally, look up the values of keys. We call this type of representation a *key-value pair*. In this case, our keys would likely be "color." When Margie comes across, say, a red sock, she looks up "red" in her line of unmatched socks. If she finds a "red" area, she might then look for additional identifiers like, say, style or hue and take things from there. Otherwise, she would create a new "red" area with the solitary red sock.

Here are how the two methods compare.* Notice that Method 1 becomes noticeably slower than Method 2 as the number of socks in the pile grows. There are many more ways of tackling the tasks in this book

* There are more nuanced ways of looking at these rates of growth. One is in terms of whether a particular method grows no faster than the rate shown (known as *big-o notation*) or no slower than the rate shown (known as *big-Ω notation*, and read *big-omega*). Another is to consider whether the rates of growth describe the best, worst, or average cases. We will be talking about these different cases later on.

Not for Distribution

than the two methods that are highlighted. The discussion is meant to emphasize two methods that are notably different in their asymptotic rates of growth, leaving out other methods whose performance may fall somewhere in between. With this scene, for instance, Margie might have chosen, alternatively, to find matches by way of the pigeonhole principle, which would have involved pulling six socks from the pile at a time and matching pairs that way.



Not for Distribution

Now, when we pick out a sock from the pile, we could probably tell fairly quickly if we've come across its matching pair. Most people's short-term memory is quite good at remembering half a dozen or so groups of things, which is what we have here. And so, coming across a sock in the pile that we've already placed to one side ought to elicit a near-immediate "Ah, I've seen that one before!" If you've ever played a card-matching game like *Memory*, the powers and limitations of that faculty ought to be familiar to you.

If we did have a bigger disparity in sock types and colors, however, our line of unmatched socks might grow quite a bit, forcing us to scan through the whole line every time we pick something new out of the pile. Scanning through a line of things, an *array*, can be time-consuming when the number of things is quite large, reason being that the thing you're looking for might be at the very end of that line. We would thus have to pass through the entirety of the array first.

Not for Distribution

9780735222120_BadChoices_TX.indd 12 2/13/17 11:59 AM

In 1953, and while working at IBM, the mathematician Hans Peter Luhn drafted an idea that gave way to an alternative structure to help alleviate the potential slowness that is inherent in array lookups. It's sometimes called an *associative array*, a *dictionary*, or a *hash table* (more salt in the wound for poor Margie). A hash table does precisely what an array does, in that it stores things in a collection, except it trades order (i.e., large black sock comes after small red sock) for near-immediate lookups, also referred to as *constant-time* lookups.*

They are called constant time because the lookup no longer depends on the length of the array. Rather, its speed is independent of the length of the array. This observation is usually true, though not always. A lot of things in software are like that, to the chagrin of many a researcher and practitioner. There are no fundamental laws in software like there are in nature. In this example, we're assuming that because of the small number of disparate socks, Margie's synapses will fire so quickly that her reaction will be near immediate.

As we will see later on, most constant-time lookups happen when we are able to model a task by way of a formula, which eliminates the need to step through a task and *iterate* over all the items at hand.† With hash tables, that formula is known as a *hash function*. Its job is to put an item into a pile and then perhaps retrieve that item at a later time fairly quickly.

All of that is just an aside, though. Our main takeaway from this scene is that an approach that reuses knowledge can be faster than one that doesn't. This is particularly useful to know when our task involves doing something over and over again—you're in a store, looking through

* In this example, Margie doesn't really care what order unmatched socks are in. All she cares about is that the socks are to one side. So the order is incidental. It is insignificant.

† For example, finding the sum of the first *n* numbers would be slower if you were to iterate over those *n* numbers one by one, summing pairs of numbers each time. It would be much faster if you were to use a formula instead: $n \times (n + 1) / 2$.

Not for Distribution

9780735222120_BadChoices_TX.indd 13 2/13/17 11:59 AM

a box of letter-shaped candles for your daughter's birthday cake. Or you're about to wash your clothes and need to separate your whites from your colors and from your delicates. Or you're trying to win a game by coming up with the longest word from a set of jumbled-up letters, as they do on the British TV show *Countdown*, where contestants have thirty seconds to come up with the longest word they can make from the nine letters in front of them.

In each of those situations, ask yourself if the task at hand could be made faster by leveraging memory—yours or the world's. With our pile of socks, we exploited the fact that we had no more than five variations of socks by maintaining a line of unmatched socks. With our box of candles, we would pick out any of the four letters we need the moment we come across it, rather than searching for "L" and then for "U" and so on.

With the dirty clothes, we might keep them in three separate baskets to begin with rather than rummage through the clothes prior to a wash. And with the longest word situation, we might look for any word that we can form at first sight and then see if we're able to lengthen it by, say, conjugating it or making it plural. In such a case, our initial choice serves as a sort of prefix (hence, memory) to subsequent words. There is a fascinating tree-based structure called a *trie* that does exactly that. It exploits the fact that words or numbers share prefixes and uses that knowledge to make things like spell-checking and auto-completing words that you might enter into a search box much faster.

Not for Distribution

9780735222120_BadChoices_TX.indd 14 2/13/17 11:59 AM

ISN'T IT INTERESTING, HOW THE MUNDANE CAN TURN
INTO SOMETHING SO ENGAGING WITH A SLIGHT SHIFT OF THE HEAD.

Not for Distribution

Not for Distribution