Limitations

- Potential for stack overflow with deep recursion. We will discuss the call stack in the next few sections.
- May be less efficient than iterative solutions in some cases.

2.2.8 What is execution?

Execution in computing can be understood through a simple cooking scenario. Let's consider making carrot soup:

You, the cook, need to:

1. Cut the carrots
2. Fill a pot with water

Now, here's where it gets interesting:

You, as the human cook, are like one CPU or executing agent. You can actively perform tasks, make decisions and process information. In this case, you can cut the carrots.

But there's another "agent" at work - the universe itself, or more specifically, gravity and the water system. This agent can also "execute tasks" for you. When you turn on the tap, gravity and the water system work to fill the pot without your constant attention.

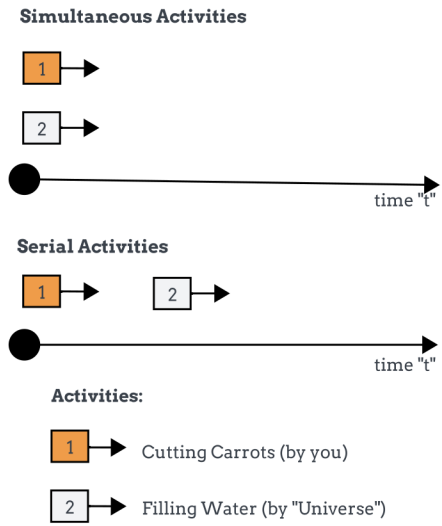

Figure 2.2: Serial and parallel (simultaneous) ordering

So now we have two "executing agents":

- You (the human/CPU) - cutting carrots
- The universe (gravity/water system) - filling the pot

By leveraging both agents simultaneously, you can achieve parallel execution:

- You start filling the pot (initiating the "universe agent")
- While the pot is filling, you cut the carrots (you, the "human agent", working)

This parallel approach is more efficient because two tasks are being completed simultaneously by different "executing agents".

In computing terms:

- You are like the main CPU
- The universe is like a separate CPU core

Execution in a computer involves several key components that are discussed in the following subsections.

2.2.9 The Execution Stack

Call Stack for Managing Function Calls

The execution stack, commonly referred to as the call stack, is a special kind of stack data structure that stores information about the active subroutines of a computer program. It manages function calls in a last-in, first-out (LIFO) manner.

It is called a "stack" because it works just like a stack in real life. Consider a stack of plates. Plates are both removed from and added to the top. Stacks are usually implemented using arrays, a type of data structure discussed in the next chapter.

When a function is called, a stack frame is created and pushed onto the stack. This frame contains the function's local variables, arguments and return address. Once the function execution is complete, the frame is popped off the stack.