Lect. 1 26/02/24
Dr. Hassanain Al-Taiy

Computational Paradigms

Overview of Computational Paradigms

Processes, Procedures, and Computers

Computer science is the study of information processes. A process is a sequence of steps. Each step changes the state of the world in some small way, and the result of all the steps produces some goal state. For example, baking a cake, mailing a letter, and planting a tree are all processes. Because they involve physical things like sugar and dirt, however, they are not pure information processes. Computer science focuses on processes that involve abstract information rather
than physical things.

A **procedure** is a description of a process. A simple process can be described just by listing the steps. The list of steps is the procedure; the act of following them is the process. A procedure that can be followed without any thought is called a mechanical procedure. An algorithm is a mechanical procedure that is guaranteed to eventually finish.

For example, here is a procedure for making coffee, adapted from the actual directions that come with a major coffeemaker:

1. Lift and open the coffeemaker lid.
2. Place a basket-type filter into the filter basket.
3. Add the desired amount of coffee and shake to level the coffee.
4. Fill the decanter with cold, fresh water to the desired capacity.
5. Pour the water into the water reservoir.
6. Close the lid.
7. Place the empty decanter on the warming plate.
8. Press the ON button.

To program computers, we need tools that allow us to describe processes precisely and succinctly. Since the procedures are carried out by a machine, every step needs to be described; we cannot rely on the operator having “common sense” (for example, to know how to fill the coffeemaker with water without explaining that water comes from a faucet, and how to turn the faucet on). Instead, we need mechanical procedures that can be followed without any thinking.

A computer is a machine that can:
1. Accept input. Input could be entered by a human typing at a keyboard, received over a network, or provided automatically by sensors attached to the computer.
2. Execute a mechanical procedure, that is, a procedure where each step can be executed without any thought.
3. Produce output. Output could be data displayed to a human, but it could also be anything that effects the world outside the computer such as electrical signals that control how a device operates.

Measuring Computing Power

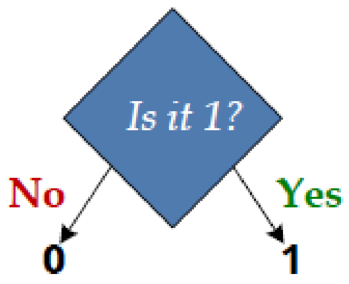
For physical machines, we can compare the power of different machines by measuring the amount of mechanical work they can perform within a given amount of time. This power can be captured with units like horsepower and watt. Physical power is not a very useful measure of computing power, though, since the amount of computing achieved for the same amount of energy varies greatly. Energy is consumed when a computer operates, but consuming energy is not the purpose of using a computer.
Two properties that measure the power of a computing machine are:
1. How much information it can process?
2. How fast can it process?

Information

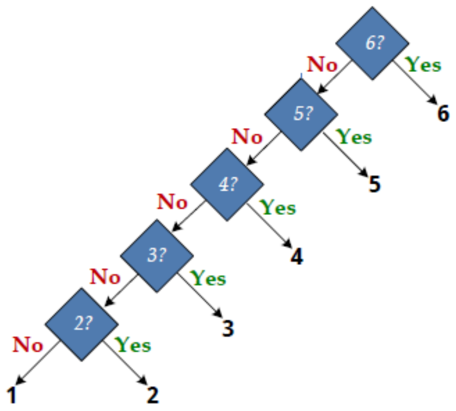
Informally, we use information to mean knowledge. But to understand information quantitatively, as something we can measure, we need a more precise way to think about information.
The way computer scientists measure information is based on how what is known changes as a result of obtaining the information. The primary unit of information is a bit. One bit of information halves the amount of uncertainty. It is equivalent to answering a “yes” or “no” question, where either answer is equally likely beforehand. Before learning the answer, there were two possibilities; after learning the answer, there is one.
We call a question with two possible answers a binary question. Since a bit can have two possible values, we often represent the values as 0 and 1.

For example, suppose we perform a fair coin toss but do not reveal the result. Half of the time, the coin will land “heads”, and the other half of the time the coin will land “tails”. Without knowing any more information, our chances of guessing the correct answer are 1

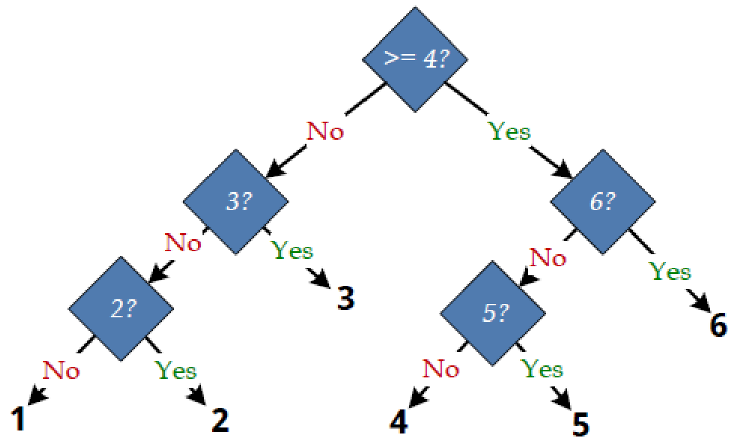
2 . One bit of information would be enough to convey either “heads” or “tails”; we can use 0 to represent “heads” and 1 to represent “tails”. So, the amount of information in a coin toss is one bit.
Similarly, one bit can distinguish between the values 0 and 1:



How many bits of information are there in the outcome of tossing a six-sided die? There are six equally likely possible outcomes, so without any more information we have a one in six chance of guessing the correct value. One bit is not enough to identify the actual number, since one bit can only distinguish between two values. We could use five binary questions like this:



Our goal is to identify questions where the “yes” and “no” answers are equally likely that way, each answer provides the most information possible. This is not the case if we start with, “Is the value 6?”, since that answer is expected to be “yes” only one time in six. Instead, we should start with a question like, “Is the value at least 4?”. Here, we expect the answer to be “yes” one half of the time, and the “yes” and “no” answers are equally likely. If the answer is “yes”, we know the result is 4, 5, or 6. With two more bits, we can distinguish between these three values (note that two bits is actually enough to distinguish among four different values, so some information is wasted here). Similarly, if the answer to the first question is no, we know the result is 1, 2, or 3. We need two more bits to distinguish which of the three values it is. Thus, with three bits, we can distinguish all six possible outcomes.



Each bit doubles the number of possibilities we can distinguish, so with three bits we can distinguish between 2* 2 *2 = 8 possibilities. In general, with n bits, we can distinguish between 2^n possibilities. Conversely, distinguishing among k possible values requires $\log_2$ k bits. The logarithm is defined such that if a = b^c then $\log_b$ a = c. Since each bit has two possibilities, we use the logarithm base 2 to determine the number of bits needed to distinguish among a set of distinct possibilities. For our six-sided die, $\log_2 6 \sim 2.58$, so we need approximately 2.58 binary questions. But, questions are discrete: we can’t ask 0.58 of a question, so we need to use three binary questions.

Units of Information

One byte is defined as eight bits. Hence, one byte of information corresponds to eight binary questions, and can distinguish among 28 (256) different values. For larger amounts of information, we use metric prefixes, but instead of scaling by factors of 1000 they scale by factors of 210 (1024). Hence, one kilobyte is 1024 bytes; one megabyte is 220 (approximately one million) bytes; one gigabyte is 230 (approximately one billion) bytes; and one terabyte is 240 (approximately one trillion) bytes.

High-Performance Computing

In high-performance computing systems, a pool of processors (processor machines or central processing units [CPUs]) connected (networked) with other resources like memory, storage, and input and output devices, and the deployed software is enabled to run in the entire system of connected components.

The processor machines can be of homogeneous or heterogeneous type. The legacy meaning of high-performance computing (HPC) is the supercomputers; however, it is not true in present-day computing scenarios. Therefore, HPC can also be attributed to mean the other computing paradigms that are discussed in the forthcoming sections, as it is a common name for all these computing systems.

Parallel Computing

Parallel computing is also one of the facets of HPC. Here, a set of processors work cooperatively to solve a computational problem. These processor machines or CPUs are mostly of homogeneous type. Therefore, this definition is the same as that of HPC and is broad enough to include supercomputers that have hundreds or thousands of processors interconnected with other resources.
One can distinguish between conventional (also known as serial or sequential or Von Neumann) computers and parallel computers in the way the applications are executed.
In serial or sequential computers, the following apply:
• It runs on a single computer/processor machine having a single CPU.
• A problem is broken down into a discrete series of instructions.
• Instructions are executed one after another.

In parallel computing, since there is simultaneous use of multiple processor machines, the following apply:
• It is run using multiple processors (multiple CPUs).
• A problem is broken down into discrete parts that can be solved concurrently.
• Each part is further broken down into a series of instructions.
• Instructions from each part are executed simultaneously on different processors.
• An overall control/coordination mechanism is employed.