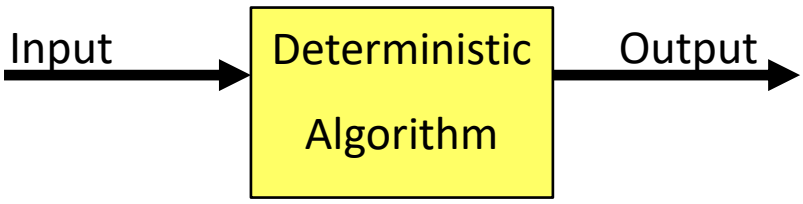
Algorithm Types

The goal of any algorithm is to solve some computational problem correctly and efficiently.

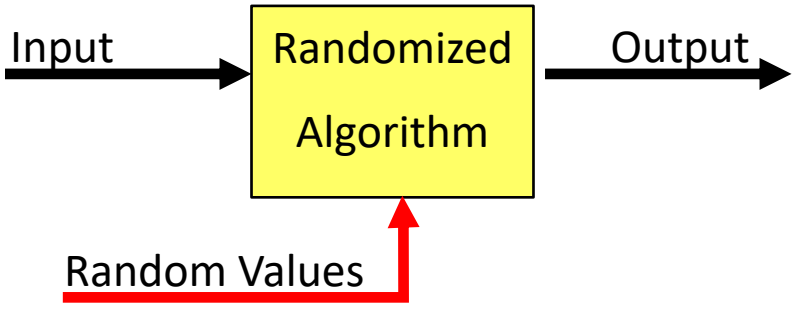
So far, this course has focused on a type of algorithm where the behavior of the algorithm is predictable from the start based on the input being processed. This type of algorithm is referred to as a deterministic algorithm. In deterministic algorithms, the next statement to execute is uniquely determined by the current statement and the state of all variables in scope on the current statement. You evaluate the current statement using the variables in scope and a unique next statement is determined. It may be a branch or jump or simply the next statement in the program.



The primary characteristic of a deterministic algorithm is that a particular input/output pair is always the same. In addition, the time and space required to transform a particular input to a particular output are always the same.

In nondeterministic algorithms there are statements at which the computer makes a choice which determines which statement and variables in scope will be executed. Dijkstra's Guarded Conditionals (see Module 3 | Content | Boon's Face-To-Face Lecture Notes Chapter 34, pp 13-14 and Module 3 | Content | Nondeterminacy and Formal Derivation of Programs) are one form of such choice statements. It is important to remember that the computer makes the choice.

Randomized algorithms can be deterministic - the next statement to execute is determined by the current statement and all variables in scope, including random variables. Randomized algorithms can be nondeterministic - the next statement to execute is a choice by the computer and the chosen statement is evaluated using all variables in scope, including random variables. The primary difference between randomized and deterministic algorithms is that the input includes a source for random numbers.



These random numbers are used to make random decisions during the execution of the algorithm. As a result, the behavior of the algorithm can vary, even on the same input.

Earlier in this course, we considered approaches to performance average-case analysis. We need to be clear here that randomized algorithms are not the same as such average-case, probabilistic analysis. In the probabilistic analyses we performed, the performance of the algorithm was assessed relative to a random sample of inputs, and the resulting performance estimation was based on the underlying distribution of that sample. The algorithms analyzed were still deterministic. In the case of the randomized algorithms, performance can vary with the same input, and we actually use the randomization to improve overall performance.