What is it?

Process of evaluating the goodness of an algorithm in various scenarios

Provides a means of reasoning about the quality of an algorithm

Provides mathematical metrics that can be used to gauge the speed of an algorithm

Why?

Provide a guide for determining when and where to use an algorithm.

Provides a classification system for determining when a particular algorithm may be best used given various characteristics of the problem.

Ensure that resources are used efficiently.

Avoid implementing a process using an algorithm that is inappropriate for the problem.

How is goodness measured?

There are many metrics.

In general, good algorithms use minimal resources to solve a problem under expected conditions

Specifically, algorithms may be classified according to the following:

* How they behave as input size increases
* How they behave for arbitrarily large input sizes

What are resources?

Time, CPU, Storage

How?

A variety of techniques are used.

Often times, algorithms are evaluated according to their performance under limiting conditions, such as arbitrarily large inputs.

What is running-time analysis?

TODO

Constant terms

In general, it is a waste of time to attempt to consider constant terms in analysis since any estimate of operations in an algorithm may differ across implementations, definition of steps, and compilation to different OS’s, etc.

It’s not appropriate to compare algorithms on factors which are architecture dependent

Ignoring constant factors yields very little loss of predictive power.

Constants ARE important as you try to optimize an algorithm, however not as part of analyzing an algorithm.

Big O

Categorizes an algorithm based on the number of operations of it’s most significant

Theta

T(n)

Worst Case

Why the focus on worst-case scenarios and asymptotic analysis?

With modern computers, it’s not interesting to try and optimize algorithms for small input sizes.

In a modern computer, small inputs will all run quickly, even if inefficient.

## Terms / Concepts

Complexity

Complexity function

Time complexity

Quantifies the amount of time it takes to run an algorithm given an input

Typically expressed in Big-O notation, removing constant factors.

Space complexity

Best case

Average case complexity

Considers the performance of an algorithm on average

Like benchmarks, requires domain knowledge in order to know what inputs are typical, most likely, and which are unlikely.

Benchmarks

Requires domain knowledge.

Worst case complexity

Usually denoted by asymptotic performance

Holds for every input of size N.

An upper bound on algorithm performance

Requires zero knowledge about the problem domain (which inputs are typical or more likely)

Appropriate for general purpose analysis and for comparing algorithms in general

Easier to apply than average-case complexity

Linear time complexity

The holy grail of algorithms

Asymptotic computational complexity

Asymptotic performance

Performance of an algorithm with arbitrarily large inputs

Asymptotic analysis focuses on comparing algorithms only under worst-case conditions.

Places low importance on the performance of an algorithm for small inputs.

As input size grows, the importance of constant factors becomes insignificant.

Conclusions don’t necessarily hold under small input sizes.

Good asymptotic performance means the complexity of an algorithm grows slowly as input size increases.

Growth rate

The degree to which the performance of an algorithm changes as input size increases

Big-O

An asymptotic metric describing the limiting behavior of an algorithm as the number of inputs increases towards infinity.

One of a family of notations used to describe the behavior of a function

Step

The unit of measure in determining the cost of an algorithm

For instance, adding two numbers may be considered a step.

Growth rates

Refers to how an algorithm’s performance changes as the input size increases

Runtime

Number of operations executed

Upper bounds

On number of operations an algorithm may perform

Fast algorithm

An algorithm whose worst case grows slowly with input size