

CSCI 311 – Algorithms and Data Structures

<http://www.eg.bucknell.edu/~csci311>

Steve Guattery

Dana 335a (department office) x7-3828

guattery@bucknell.edu

<http://www.eg.bucknell.edu/~guattery>

Ben Razet

Dana 318 x7-3397

benoit.razet@bucknell.edu

<http://www.eg.bucknell.edu/~bhcr001>

This course does not have a lab, but it does have a recitation that meets Thursday afternoons.

We **will** meet tomorrow. Assignment:

Read Appendix A and Section 3.2 in the text.

What is a *data structure*?

“A data structure consists of a base storage method (e.g., an array) and one or more algorithms used to access or modify that data.”

“A *data structure* is how information is stored in the computer; a *data type* is a data structure plus its operations.”

Key point: Data structures determine how data is stored on a computer. They are part of an *implementation*.

What is an *algorithm*?

“A sequence of computational steps that transform the input to the output.”

“A problem-solving method *suitable for* implementation as a computer program.”

Key points: Algorithms are not (necessarily) implementations. They are often described in terms of *mathematical objects* (e.g., ordered array, tree, graph) that may or may not immediately correspond to a data structure.

What is an *abstract data type* (ADT)?

An ADT specifies the data and operations that will be used in a data type, but does not specify the implementation.

ADTs serve as a bridge from an algorithmic description to an implementation using data structures.

Process for creating a program:

Real-world problem on real-world objects



Mathematical representation of problem and objects + algorithm(s)



Computer implementation on data structures

The Big Question for This Course:

Resource usage of algorithms and data structures.

What resources do they use?

Resources for algorithms and data structures:

- Time (instructions/machine cycles)
- Space (memory)

We will focus mainly on time.

We want algorithmic comparisons to be:

- independent of processor (*abstract step*)
- independent of implementation (*ignore start-up overhead and constants*)
- related to how the resource depends on problem size (*asymptotic analysis*)