
**COURSE DESCRIPTION**

1. **Concepts, notation, and terminology** for reasoning quantitatively about the **efficiency** of algorithms
2. **Important data-structures and algorithms**
3. **Fundamental techniques** for algorithm design
Administrative matters

COURSE WEB-PAGE

• **Monitor frequently!** – assignments, test dates, etc. announced here
• Accessible off dept web-page ([Academics -> Course home-page links](http://www.cs.unc.edu/~baruah/Teaching/2016-1Sp/))

GRADEs are maintained on sakai

Piazza for discussions

<<Tour of [course web-page]>>
Administrative matters

READING ASSIGNMENTS

• Sections of the text – on the course web-site
• You **must** read these – covered in the tests
• May discuss interactively at the start of the next class

PROBLEM ASSIGNMENTS

• Not graded, but covered in the tests
• Suggestion: form study-groups

ATTENDANCE

• You are expected to attend most lectures (although no roll call)
• Occasional pop-quizzes for extra credit

<<Roll call and background survey>>
Topics to be covered

• Introduction. The role of algorithms in computer science
• Asymptotic notation
• Solving recurrences
• Sorting and Order statistics
• Search structures: red-black trees
• Introduction to graphs
• Algorithm design: Dynamic Programming
• Algorithm design: the Greedy strategy
• NP-Completeness
• Linear Programming

1. Concepts, notation, and terminology for reasoning quantitatively about efficiency of algorithms
2. Important data-structures and algorithms
3. Fundamental techniques for algorithm design
Introduction to Algorithms
What is an algorithm?

Informally, an *algorithm* is any well-defined computational procedure that takes some value, or set of values, as *input* and produces some value, or set of values, as *output*. An algorithm is thus a sequence of computational steps that transform the input into the output.

We can also view an algorithm as a tool for solving a well-specified *computational problem*. The statement of the problem specifies in general terms the desired input/output relationship. The algorithm describes a specific computational procedure for achieving that input/output relationship.
Why study algorithms?

**Important** in all other branches of computer science

- Sec 1.1 – Internet; e-commerce; manufacturing; cryptography

Prime **enabler** of innovation

- E.g., search algorithms (pagerank)
- Read Sec 1.2

“Everyone knows Moore’s Law – a prediction made in 1965 by Intel co-founder Gordon Moore that the density of transistors in integrated circuits would continue to double every 1 to 2 years....in many areas, performance gains due to improvements in algorithms have vastly exceeded even the dramatic performance gains due to increased processor speed.”

- Excerpt from *Report to the President and Congress: Designing a Digital Future*, December 2010 (page 71).
Why study algorithms?

**Important** in all other branches of computer science

Sec 1.1 – Internet; e-commerce; manufacturing; cryptography

Prime **enabler** of innovation

E.g., search algorithms (pagerank)

Read Sec 1.2

Ideas have been **applied to other domains**

  e.g., economics – auctions and mechanisms

Develops **problem-solving skills**

Often **fun** (for some of us)