CS 200: Algorithm Analysis

Instructor: Dr. Heather Guarnera

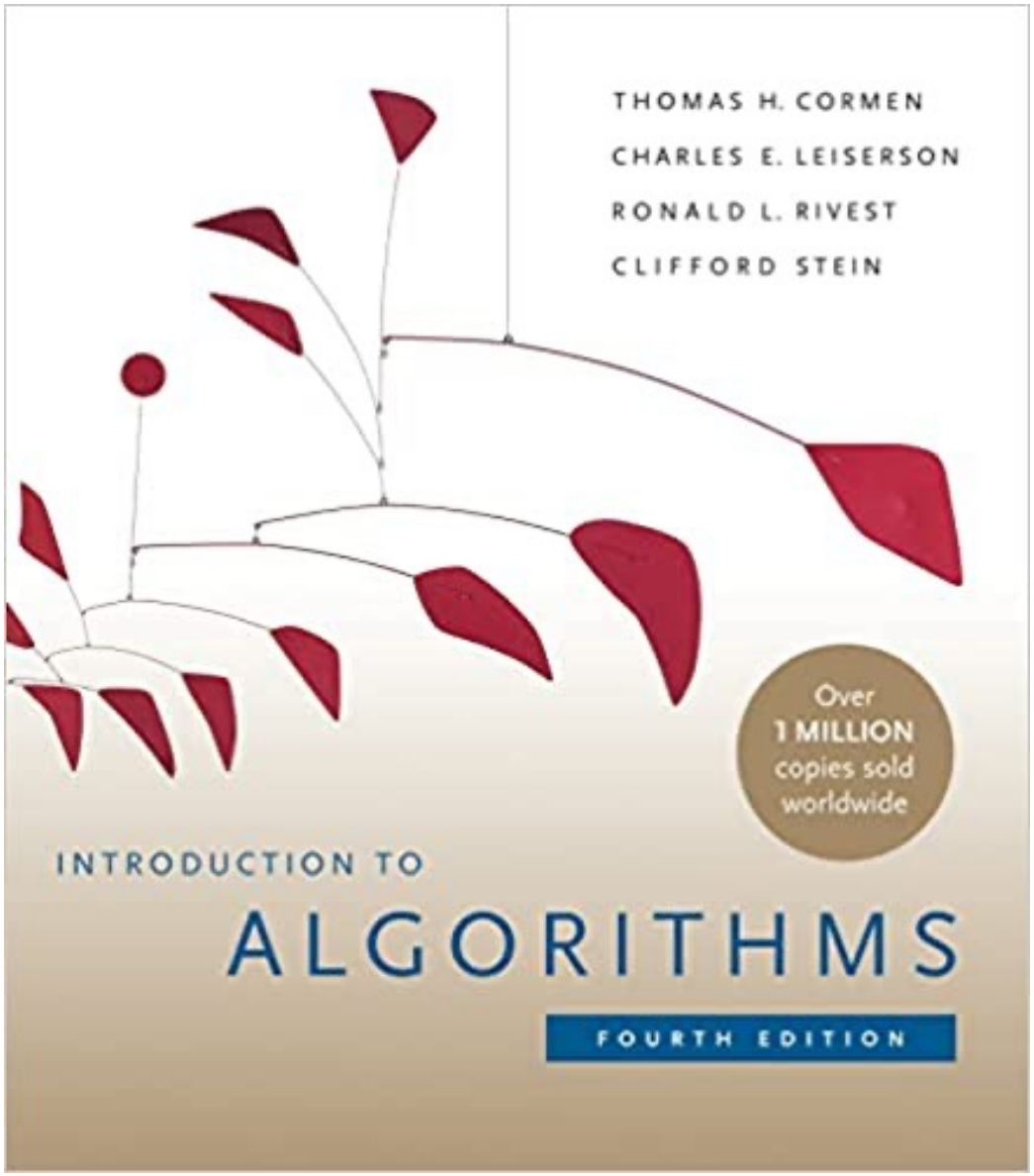
TAs: TBA

Administrative info:

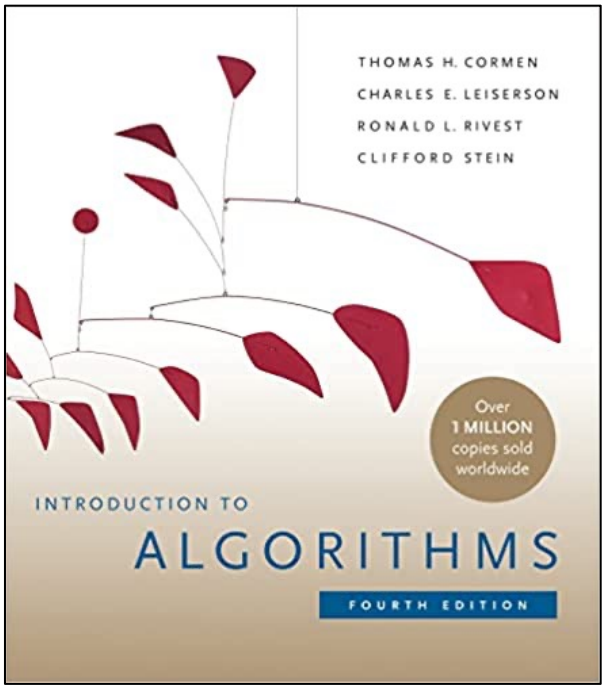
- Course website
- Book
- Syllabus
- Moodle

This course serves two purposes:

- Design and analyze algorithms
- Prepare for Senior IS

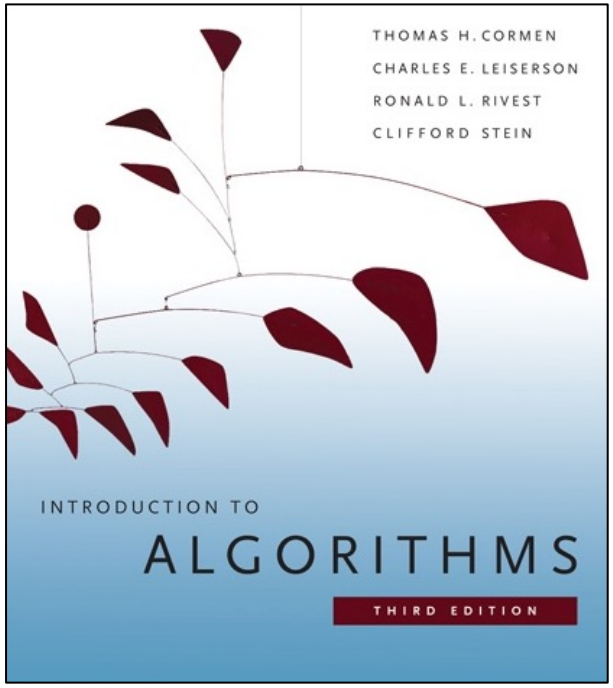


Versions of the book…

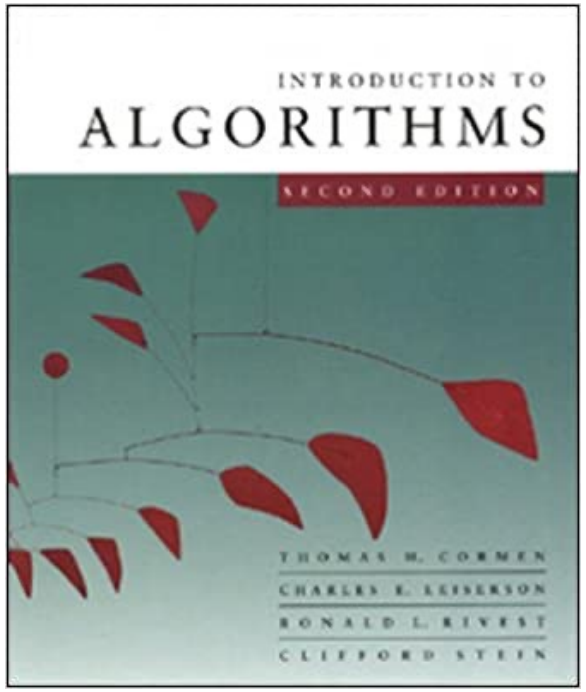


4th edition
We are using this one

It is available **online for free** through College of Wooster libraries



3rd edition



2nd edition

Introduction

CLRS 1.1 & 1.2

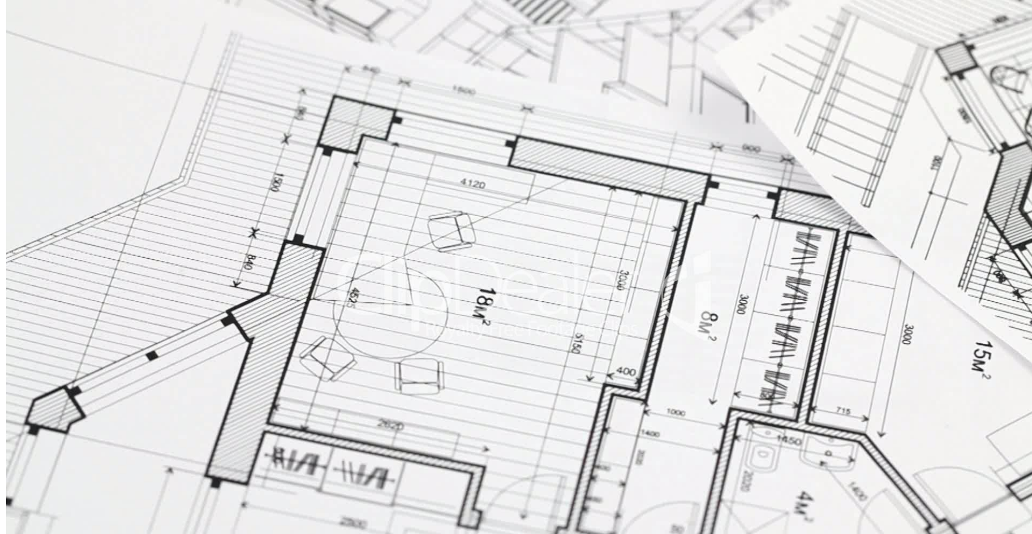
Example: Boss assigns a task

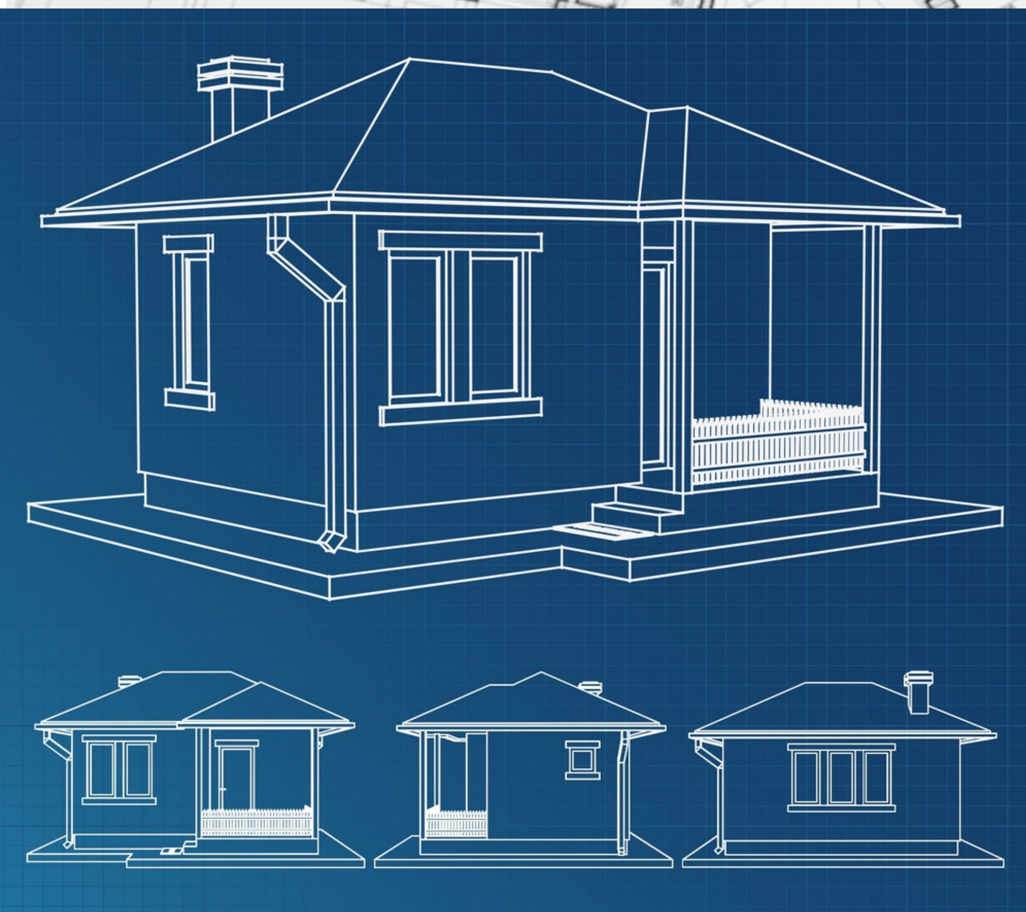
- Given today's prices of pork, grain, sawdust, etc…
- Given constraints on what constitutes a hotdog.
- Make the cheapest hotdog.

Every industry asks these questions.

- Mundane programmer: "Um? Tell me what to code."
- Better: "I learned an algorithm that will work."
- Best: "I can develop an algorithm for you."

How to do this?



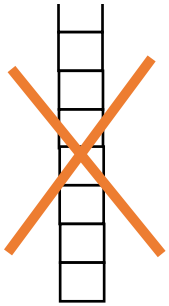



PETER RUPRECHT

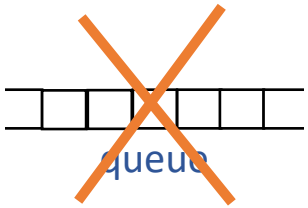
Tools you need

Example: Design an inventory system which can quickly find an item.

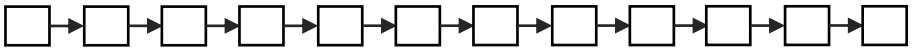
- What data structure to use?



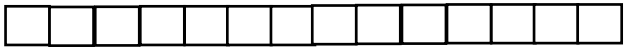
stack



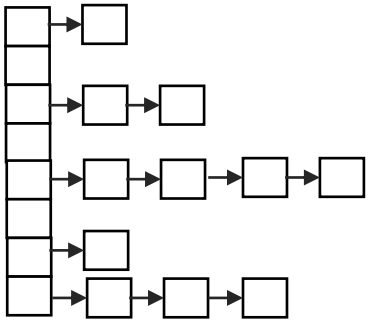
queue



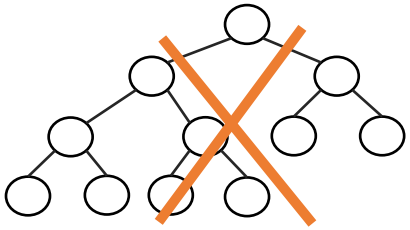
linked list



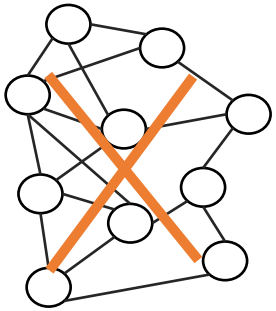
array



hash table



tree



graph

Tools you need

Example: Design an inventory system which can quickly find an item.

- What approach to take?

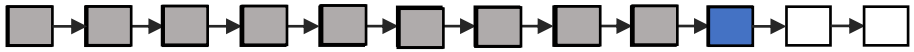
  Brute force

  Dynamic programming

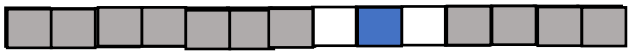
  Divide and conquer

  Greedy method

  Prune and search

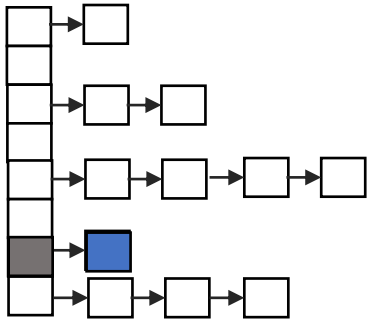


linked list



array

- Are there any existing algorithms that could be used/modified?



hash table

Tools you need

Example: Design an inventory system which can quickly find an item.

- How to determine which solution is best?
- Does it work as required?

  Rationalization

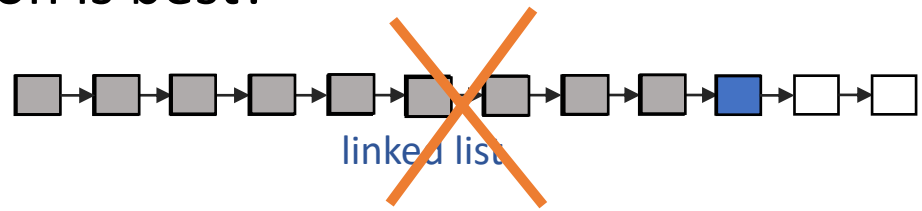
  Proof of correctness

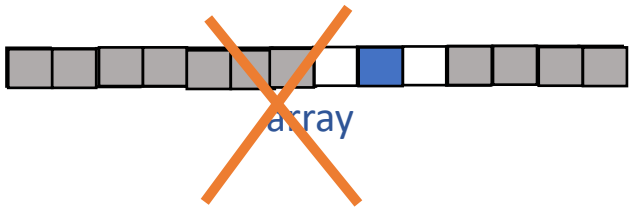
- How much memory is required? How long does it take?

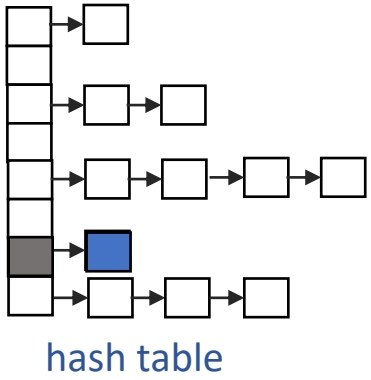
  Big-oh notation

  Amortization

  Complexity analysis







Algorithm Analysis

- How to evaluate algorithms (correctness, complexity)
  - Notations and abstractions for describing algorithms
- Advanced data structures and their analysis
- Fundamental techniques to solve the vast array of unfamiliar problems that arise in a rapidly changing field
  - Up to date grasp of fundamental problems and solutions
  - Approaches to solve
- To think algorithmically like a ‘real’ computer scientist

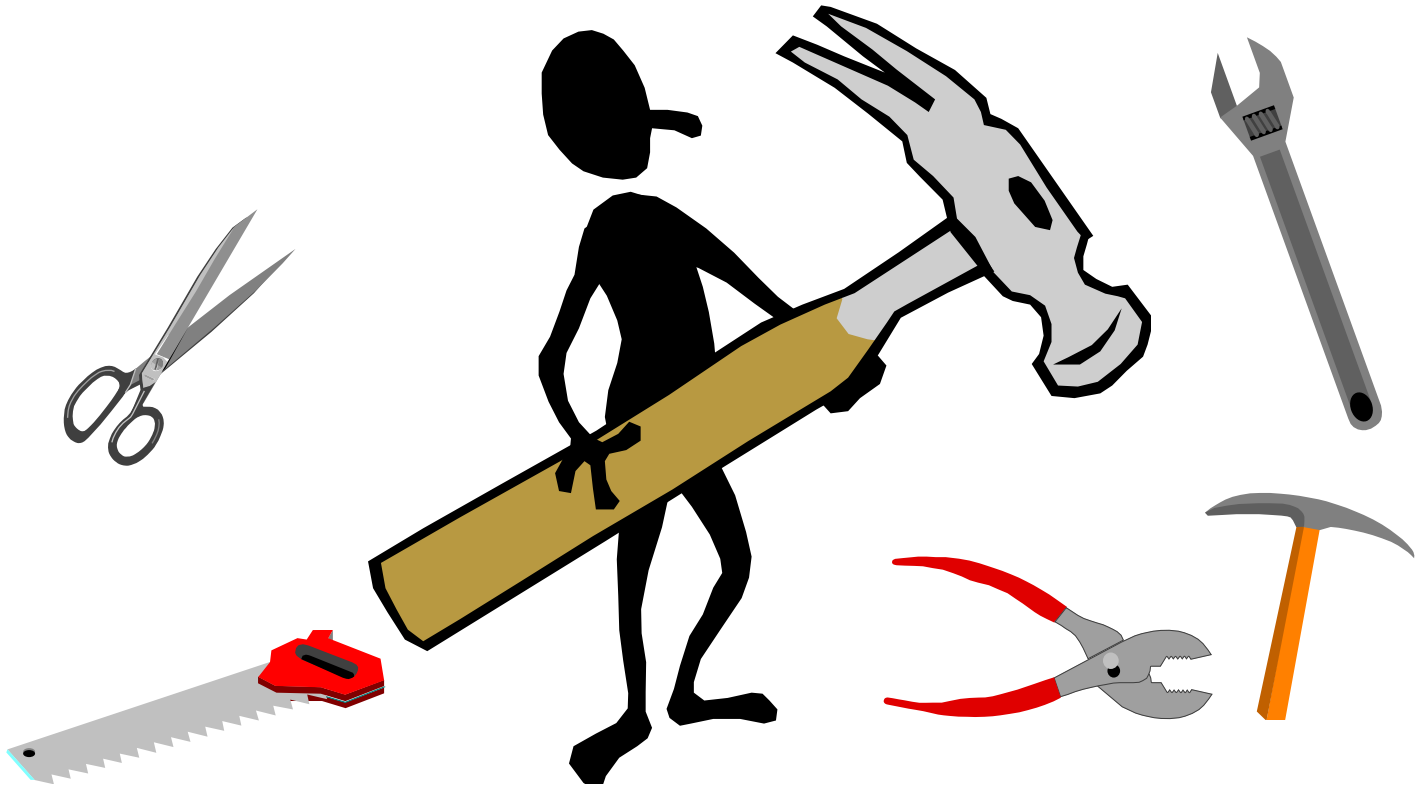
Course Content

- A list of algorithms
  - Learn the code
  - Trace them until you are convinced that they work
  - Implement them.

```
class InsertionSortAlgorithm extends SortAlgorithm {
void sort(int a[]) throws Exception {
        for (int i = 1; i < a.length; i++) {
            int j = i;
            int B = a[i];
            while ((j > 0) && (a[j-1] > B)) {
              a[j] = a[j-1];
              j--;  }
            a[j] = B;
        }
}
```

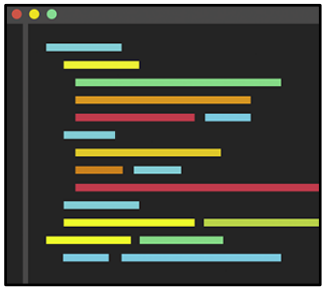
Course Content

- A survey of algorithmic design techniques
- Abstract thinking
- How to develop new algorithms for any problem that may arise



Start with some math

Time complexity as a function



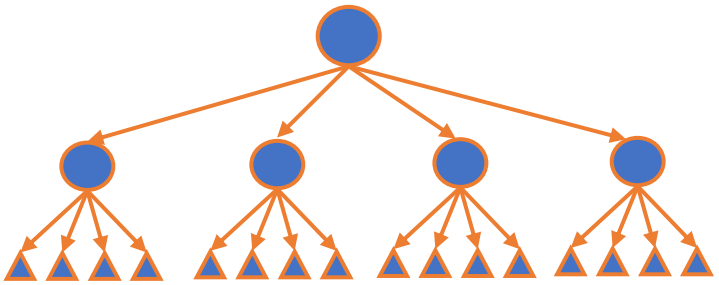
$t(n) = \Theta(n^2)$

Counting primitive operations

- Sequences and summations
- Linear functions
- Logarithmic and exponential functions

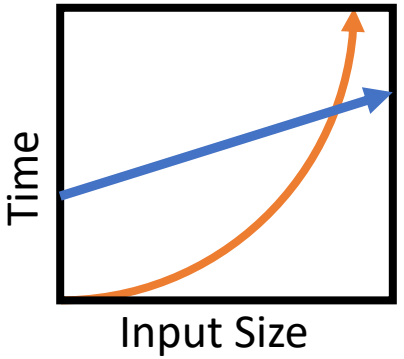
$$a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} = \sum_{k=0}^{n-1} ar^k = a\left(\frac{1-r^n}{1-r}\right)$$

Recurrence Relations

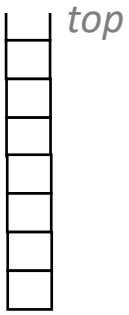


$T(n) = a\,T(n/b) + f(n)$

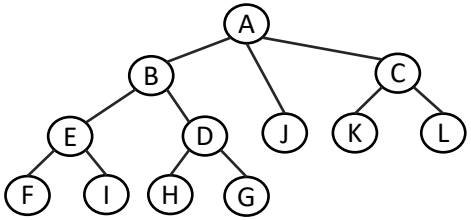
Classifying functions



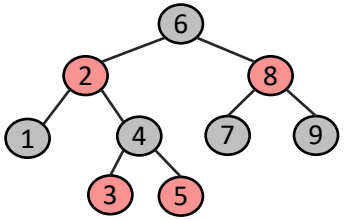
Data Structures



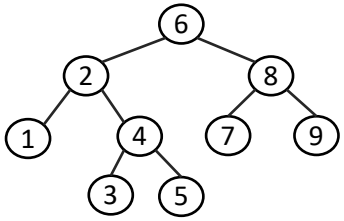
stack



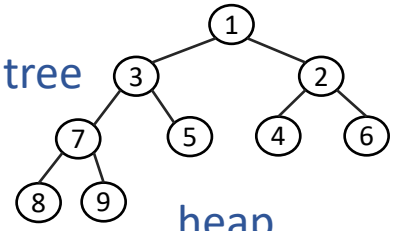
tree



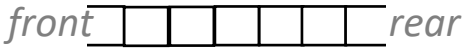
red black tree



binary search tree

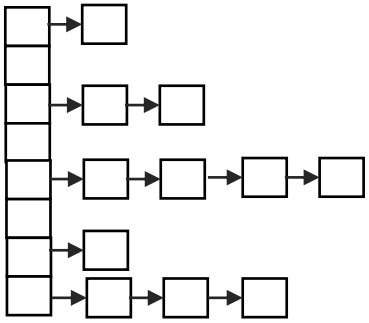


heap
&
priority queues

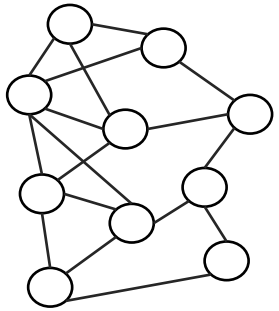


queue

linked list



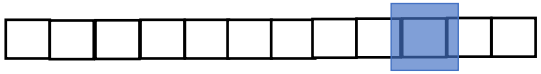
hash table
&
dictionaries

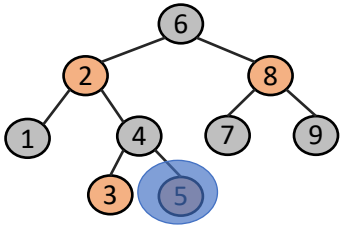


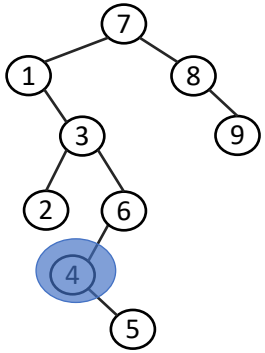
graph

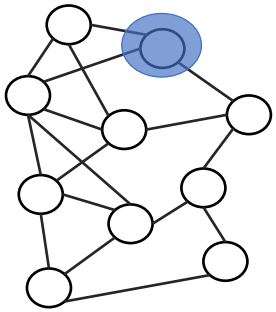
vector

Searching & Sorting









insertion sort

selection sort

heap sort

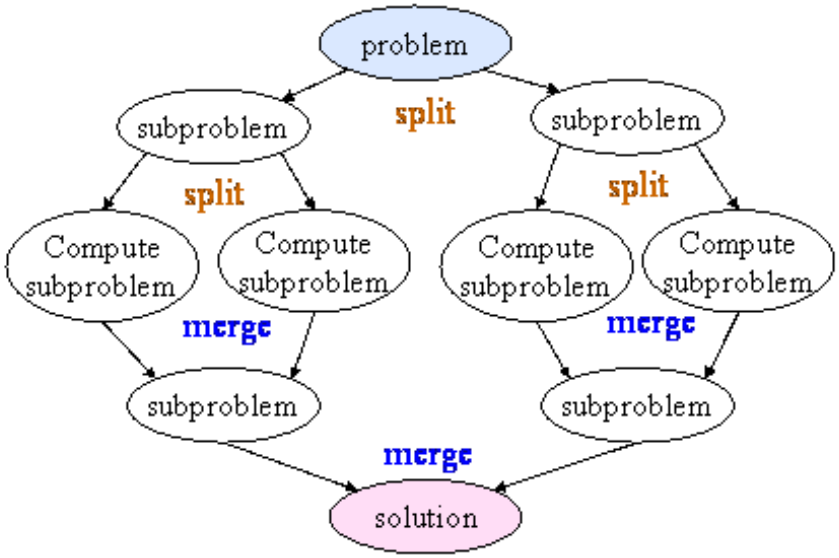
merge sort

quick sort

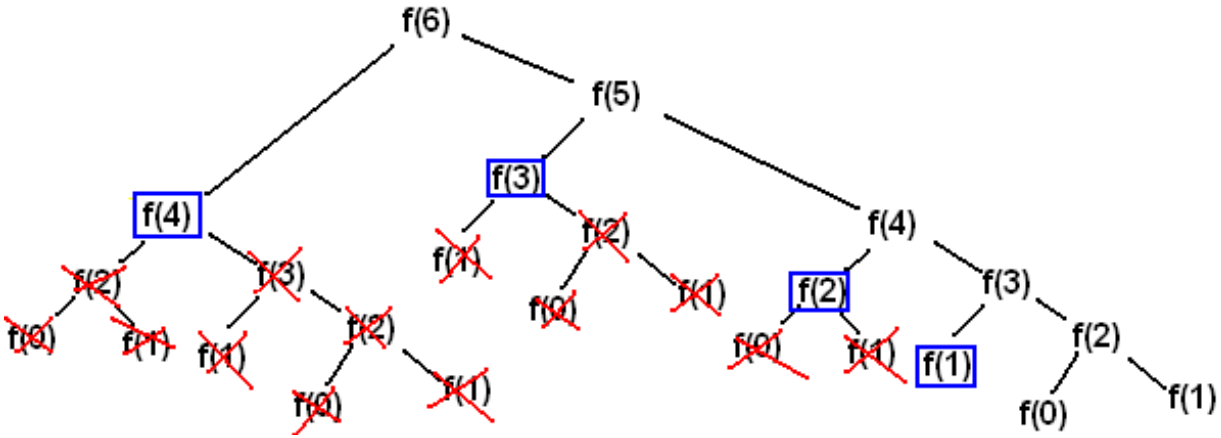
Fundamental Techniques

Greedy Algorithms

Divide and Conquer

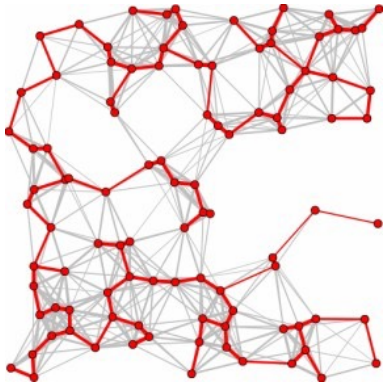


Dynamic Programming

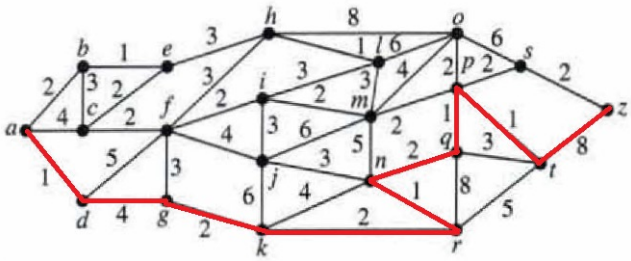


Graph algorithms

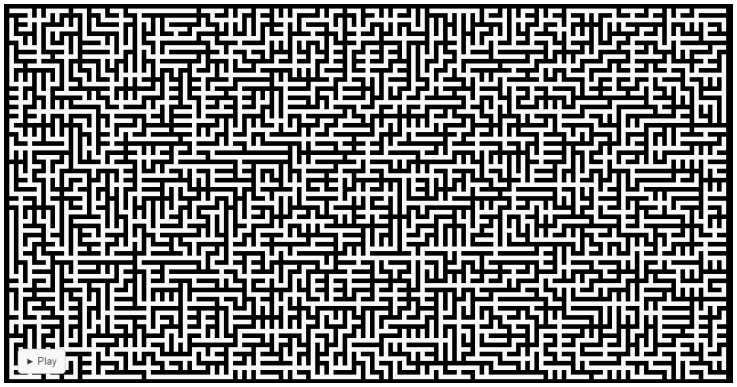
Minimum Spanning Tree



Shortest path



Graph search



Useful Learning Techniques

- You are expected to read ahead (before class)
  - This will facilitate more productive discussion during class
  - Plicker questions will be based on assigned reading
- Guess at potential algorithms for solving a problem
  - Look for input instances where your algorithm is wrong
- Practice explaining
  - You'll be tested on your ability to explain material
- Ask questions
  - Why is it done this way and not that way?

Applications in a wide variety of areas: Junior IS

Each student will undertake a major individual computer science project in the context of a particular application of interest to the student.

- Written component, software component, presentation
- Should include algorithm analysis
- Cover methods/topics not covered

The purpose is to prepare you for the process of Senior IS

Some previous topics have included:

- Image / handwriting recognition
- P vs. NP
- Path finding applications
- Procedural generation