Introduction to Computer Science

CSCI 109

Readings

St. Amant, Ch. 4

Andrew Goodney

Fall 2019

"An **algorithm** (pronounced AL-go-rith-um) is a procedure or formula for solving a problem. The word derives from the name of the mathematician, Mohammed ibn-Musa al-Khwarizmi, who was part of the royal court in Baghdad and who lived from about 780 to 850."

Reminders

- First quiz today
- Please take the survey if you haven't already.
  - https://tinyurl.com/y2cot2r5
  - Password: CS109Fall2019

Where are we?

Date	Topic		Assigned	Due	Quizzes/Midterm/Final	Slide Deck
26-Aug	Introduction	What is computing, how did computers come to be?				1
2-Sep	Labor day					
9-Sep	Computer architecture	How is a modern computer built? Basic architecture and assembly	HW1			2
→ 16-Sep	Data structures	Why organize data? Basic structures for organizing data			**Quiz 1** on material taught in class 8/26 and 9/9	3
23-Sep	Data structures	Trees, Graphs and Traversals	HW2	HW1		4
30-Sep	More Algorithms/Data Structures	Recursion and run-time				5
7-Oct	Complexity and combinatorics	How "long" does it take to run an algorithm. P vs NP			**Quiz 2** on material taught in class 9/16 and 9/23	5
14-Oct	Algorithms and programming	Programming, languages and compilers		HW2	**Quiz 3** on material taught in class 9/30	7
21-Oct	Operating systems	What is an OS? Why do you need one?	HW3		**Quiz 4** on material taught in class 10/7	8
28-Oct	Midterm	Midterm			**Midterm** on all material taught so far.	
4-Nov	Computer networks	How are networks organized? How is the Internet organized?		HW3		9
11-Nov	Artificial intelligence	What is AI? Search, plannning and a quick introduction to machine learning			**Quiz 5** on material taught in class 9/4	10
18-Nov	The limits of computation	What can (and can't) be computed?	HW4		**Quiz 6** on material taught in class 11/11	11
25-Nov	Robotics	Robotics: background and modern systems (e.g., self-driving cars)			**Quiz 7** on material taught in class 11/18	12
2-Dec	Summary, recap, review	Summary, recap, review for final		HW4	**Quiz 8** on material taught in class 11/25	13
13-Dec	Final exam 11 am - 1 pm in SGM 123				**Final** on all material covered in the semester	

Data Structures and Algorithms

- A problem-solving view of computers and computing
- Organizing information: sequences and trees
- Organizing information: graphs
- Abstract data types: recursion

Reading:
St. Amant Ch. 4

- “The architecture level gives us a very detailed view of what happens on a computer. But trying to understand everything a computer does at this level would be…(insert analogy about perspective). If all we can see is fine detail, it can be hard to grasp what’s happening on a larger scale.”

- “Here’s a different perspective: computers solve problems. Solving problems, in contrast to executing instructions, means not having to worry about all the details at once. Instead, we can think in more abstract terms. How should we represent a problem? Can we break a problem down into smaller pieces so that it’s easier to solve? What would a solution procedure look like, in the abstract?

- ”Answering these questions is a matter of representation. We’ve already seen representation, in the encoding of data and instructions in a form that’s convenient for a computer. Now we need to think more generally about how to represent problems and their solutions.” – st. Amant pg. 52

- When thinking about solving problems with computers (somewhat due to the nature of computers), three abstract data types are essential:
  - Sequences
  - Trees
  - Graphs
- Part of the course is essentially an extended vocabulary lesson
  - So you’re prepared to understand and learn these topics in detail in other courses

Problem Solving

- Architecture puts the computer under the microscope
  - Imagine solving *all* problems by thinking about the computer at the architecture level
- Early computer scientists *had* to do this
  - Luckily we don’t.

Problem Solving

- Computers are used to solve problems
- Abstraction for problems
  - How to represent a problem ?
  - How to break down a problem into smaller parts ?
  - What does a solution look like ?
- Two key building blocks
  - Abstract data types
  - Algorithms

Algorithms

- Algorithm: a step by step description of actions to solve a problem
- Typically at an abstract level
- Analogy: clearly written recipe for preparing a meal
- More in the next few lectures

"Algorithms are models of procedures at an abstract level we decided is appropriate." [St. Amant, pp. 53]

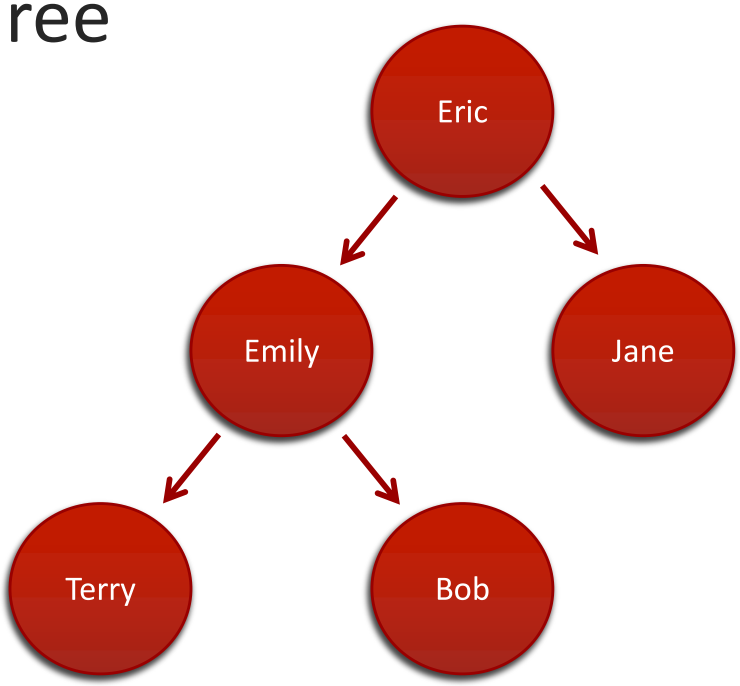
Abstract Data Types

- Models of collections of information
  - Chosen to help solve a problem
- Typically at an abstract level
  - Don’t deal with implementation details: memory layout, pointers, etc.

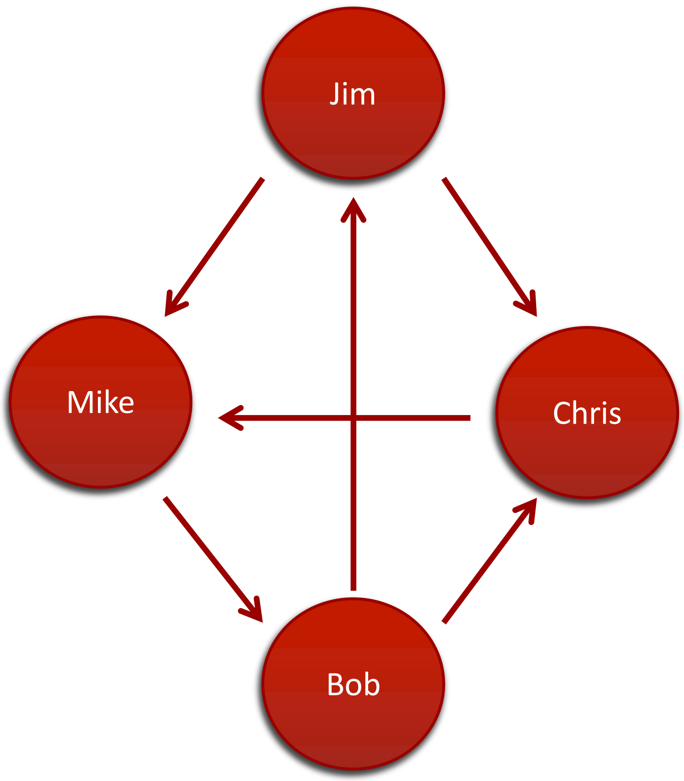
“… describes what can be done with a collection of information, without going down to the level of computer storage.” [St. Amant, pp. 53]

Sequences, Trees and Graphs

- Sequence: a list
  - Items are called elements
  - Item number is called the index
- Tree



- Graph



Motivation for Abstract Data Structures

- The nature of some data, and the way we need to accesses it often requires some structure, or organization to make things efficient (or even possible)
- Data: large set of names (maybe attendance data)
- Problems: did Jelena attend on 9/9? How many lectures did Mario attend? Which students didn't attend 8/26?

Is ‘Jelena’ on this list?

- Byron
- Therese
- Alpha
- Christopher
- Jacquelyn
- Amada
- Araceli
- Deanna
- Mario
- Pamela
- Lin
- Hester

- Lenora
- Staci
- Emma
- Elsa
- Derrick
- Kelley
- Kathe
- Mohammad
- Julia
- Renda
- Kylee
- Keren

- Jayna
- Joy
- Sean
- Basilia
- Cassie
- Sharice
- Carina
- Liv
- Clara
- Bess
- Simone
- Michiko
- Elmer

- Jayna
- Jesusa
- Dion
- Orpha
- Denice
- Tad
- Geraldine
- Bradley
- Mariah
- Lyndsey
- Marcia
- Beatrice
- Keri
- Thu

Option #1 No Data Structure

- Store names in the computer with no organization
- Scan all of them every time a question is asked

Is 'Lilly' on this list?

- Allene
- Berenice
- Bernadine
- Candelaria
- Carli
- Carry
- Chau
- Cynthia
- Clement
- Davina
- Exie
- Ezequiel
- Filiberto
- Francisca
- Fred
- Gayle
- Gudrun
- Huey
- Isaiah
- Janey
- Jen
- Joanne
- Joanie
- Laney
- Lenora
- Lilliam
- Lilly
- Lina
- Lorinda
- Lulu
- Michelle
- Madelaine
- Marielle
- Mauro
- Mayola
- Mikaela
- Pamala
- Pinkie
- Princess
- Rocco
- Rosanne
- Sally
- Season
- Sidney
- Tamica
- Tilda
- Val
- Vinita
- Yaeko
- Yoshiko

Option #2 Sorted List

- Store names in sorted order
- This implies **structure** to the data
- Also, if the names start out un-sorted, how do we get to sorted state?

Sequences aka Lists

- Sequences are our first fundamental data structure
- Sequences hold items
  - Items = what ever we need. It's abstract.
- Sequences have the notion of order
  - Items come one after another
- Sequences can be accessed by index, or relative
  - Find the 5^{th} item
  - Or move to next or previous from current item
- The "how" (implementation) is not important (now)
  - Arrays (C, C++), Vectors (C++), ArrayList (Java), Lists (Python)…
  - These are all different implementations of this abstract data structure

Sequence Tasks

- Most "questions" (problems) that are solved using sequences are essentially one of two questions:
- Is item A in sequence X?
- Where in sequence Y is item B?
- Both of these are answered by searching the sequence

Sequences: Searching

- Sequential search: start at 1, proceed to next location…
- If names in the list are *sorted* (say in alphabetical order), then how to proceed?
  - Start in the 'middle'
  - Decide if the name you're looking for is in the first half or second
  - 'Zoom in' to the correct half
  - Start in the 'middle'
  - Decide if the name you're looking for is in the first half or second
  - 'Zoom in' to the correct half
  - …
- Which is more efficient (under what conditions)?

brute force ↑

divide-and-conquer ↑

Sorting

- If searching a sorted sequence is more efficient (per search), this implies we need a way to sort a sequence!
- Sorting algorithms are fundamental to CS
  - Used A LOT to teach various CS and programming concepts
- Computer Scientists like coming up with better more efficient ways to sort data
  - Even have contests!
- We'll look at two algorithms with very different designs
  - Selection Sort
  - Quick Sort

Sorting: Selection Sort

- Sorting: putting a set of items in order
- Simplest way: selection sort
  - March down the list starting at the beginning and find the smallest number
  - Exchange the smallest number with the number at location 1
  - March down the list starting at the second location and find the smallest number (overall second-smallest number)
  - Exchange the smallest number with the number at location 2
  - …

Sorting: Selection Sort

13 4 3 5 12 6 20 10

3 4 13 5 12 6 20 10

3 4 13 5 12 6 20 10

3 4 5 13 12 6 20 10

3 4 5 6 12 13 20 10

3 4 5 6 10 13 20 12

3 4 5 6 10 12 20 13

3 4 5 6 10 12 13 20

9 8 6 5 3 1

1 8 6 5 3 9

1 3 6 5 8 9

1 3 5 6 8 9

1 3 5 6 8 9

1 3 5 6 8 9

3 6 7 9 10 20 1

1 6 7 9 10 20 3

1 3 7 9 10 20 6

1 3 6 9 10 20 7

1 3 6 7 10 20 9

1 3 6 7 9 20 10

1 3 6 7 9 10 20

Sorting: Selection Sort

- Sorting: putting a set of items in order
- Simplest way: selection sort
  - March down the list starting at the beginning and find the smallest number
  - Exchange the smallest number with the number at location 1
  - March down the list starting at the second location and find the smallest number (overall second-smallest number)
  - Exchange the smallest number with the number at location 2
  - …
- How long does this take? Can we do it faster?
- Yes, use divide-and-conquer

How long does it take?

- We just asked an interesting question, did you notice?
- ”How long does it take?”
- This question might (should) bother some of you.
  - Why?

How long does it take?

- WTH are we even asking here?
- We’re working with an “abstract data type”
- What does ”time” even mean?
- We need to abstract time as well!
- Given some data “how long does it take” = how much “work” do we do
- “Work”
  - Operations like moving an item, copying and item, comparing two items
  - Abstract steps required
- We’ll spend a lot more time discussing this over the next few lectures

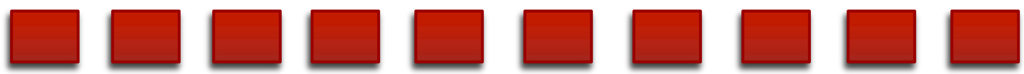
Sorting: Quicksort

- Pick a 'middle' element in the sequence (this is called the pivot)
- Put all elements smaller than the pivot on its left
- Put all elements larger than the pivot on the right
- Now you have two smaller sorting problems because you have an unsorted list to the left of the pivot and an unsorted list to the right of the pivot
- Sort the sequence on the left (use Quicksort!)
- Sort the sequence on the right (use Quicksort!)

Sorting: Quicksort

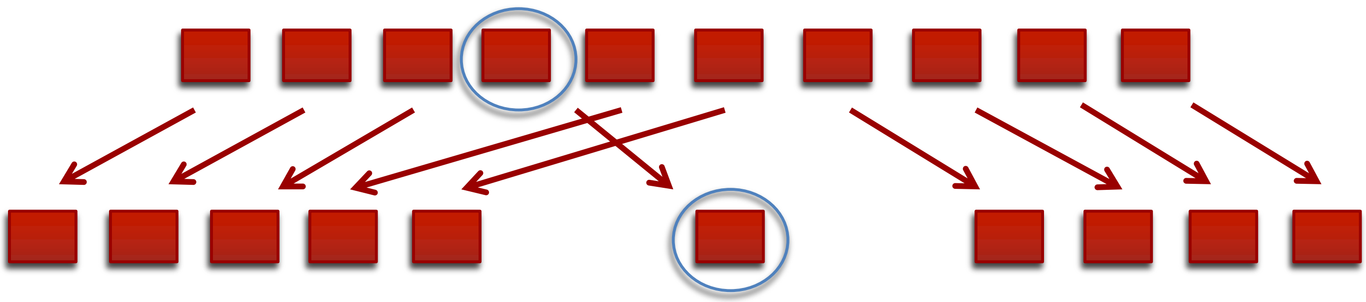
- Pick a ‘middle’ element in the sequence (this is called the pivot)
- Put all elements smaller than the pivot on its left
- Put all elements larger than the pivot on the right
- Now you have two smaller sorting problems because you have an unsorted list to the left of the pivot and an unsorted list to the right of the pivot
- Sort the sequence on the left (use Quicksort!)
  - Pick a ‘middle’ element in the sequence (this is called the pivot)
  - Put all elements smaller than the pivot on its left
  - Put all elements larger than the pivot on the right
  - Now you have two smaller sorting problems because you have an unsorted list to the left of the pivot and an unsorted list to the right of the pivot
  - Sort the sequence on the left (use Quicksort!)
  - Sort the sequence on the right (use Quicksort!)
- Sort the sequence on the right (use Quicksort!)
  - Pick a ‘middle’ element in the sequence (this is called the pivot)
  - Put all elements smaller than the pivot on its left
  - Put all elements larger than the pivot on the right
  - Now you have two smaller sorting problems because you have an unsorted list to the left of the pivot and an unsorted list to the right of the pivot
  - Sort the sequence on the left (use Quicksort!)
  - Sort the sequence on the right (use Quicksort!)

Quicksort



This is an unsorted list (e.g., a list of numbers not in order)

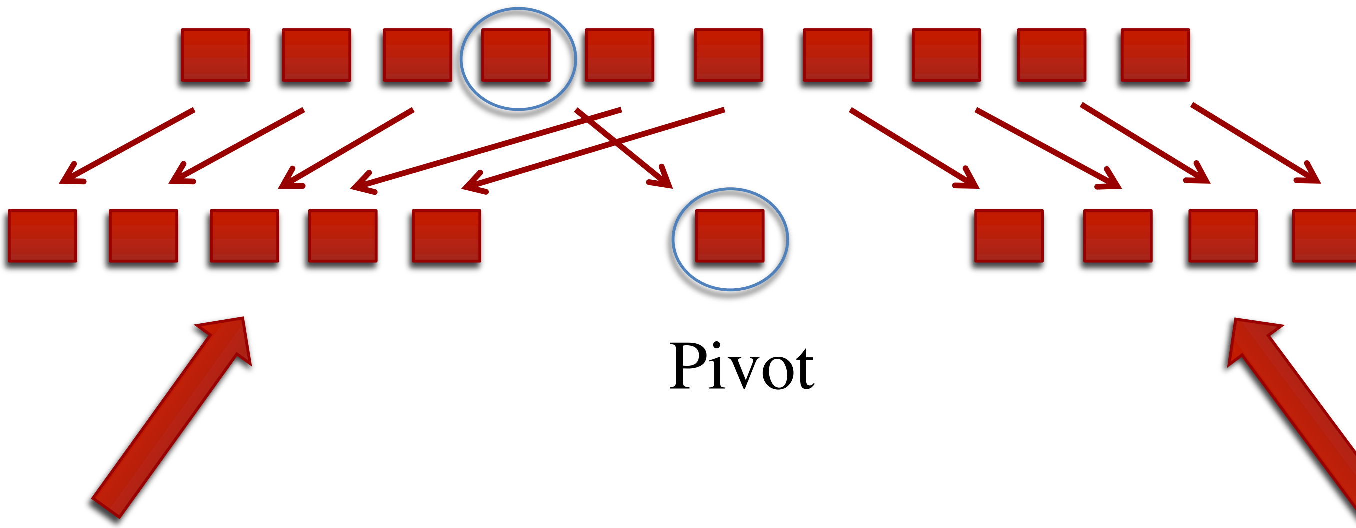
Quicksort



Pivot

Choose a pivot and put all elements smaller than the pivot
to the left of the pivot and all elements larger than the pivot to its right

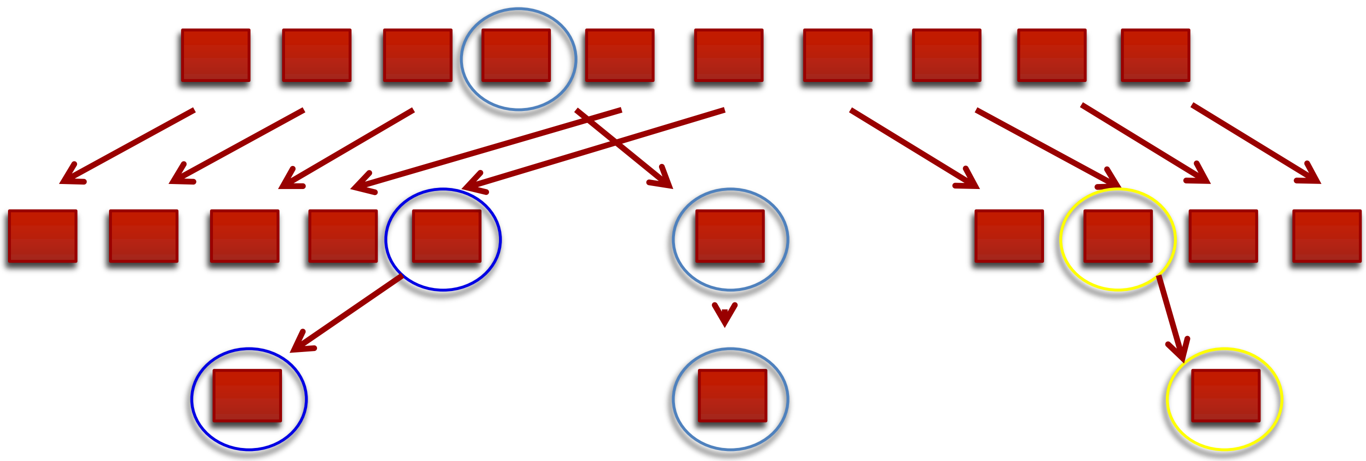
Quicksort



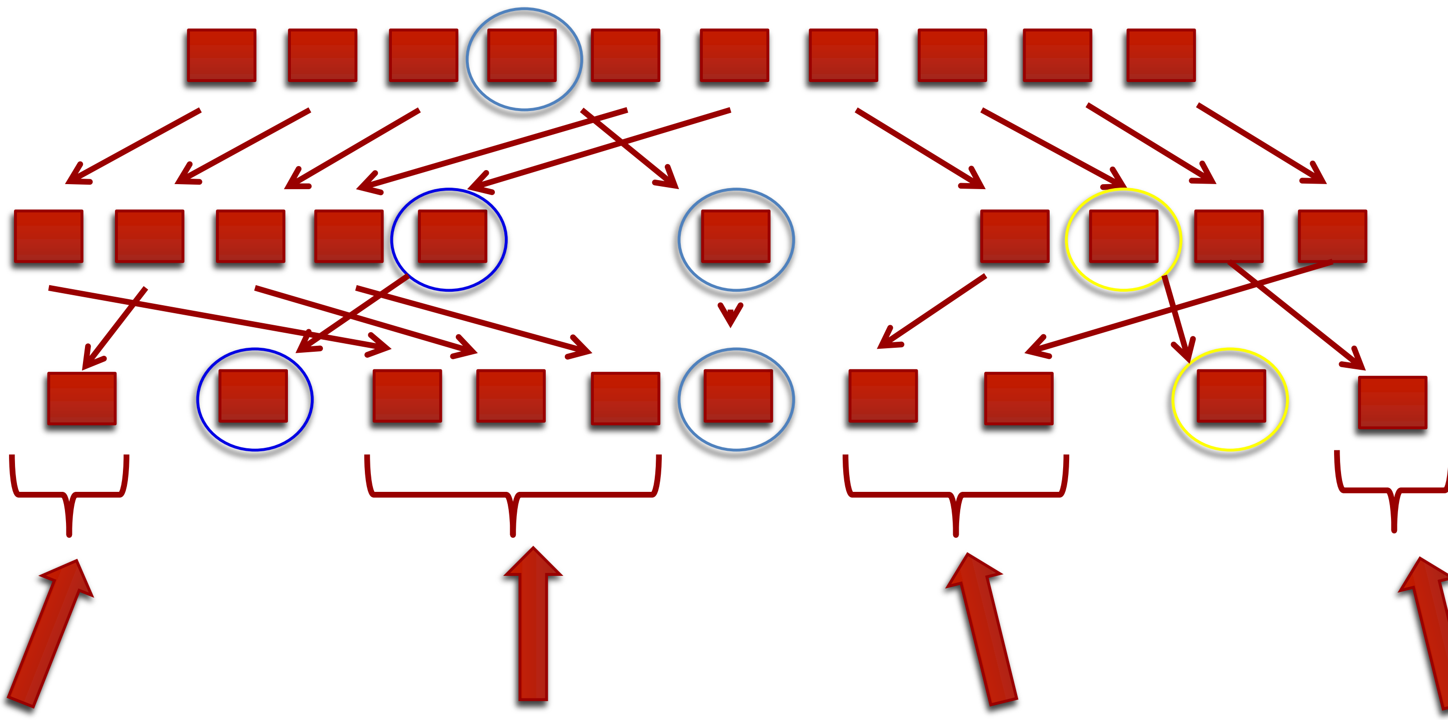
This is an unsorted list of all elements smaller than the pivot

This is an unsorted list of all elements larger than the pivot

Quicksort

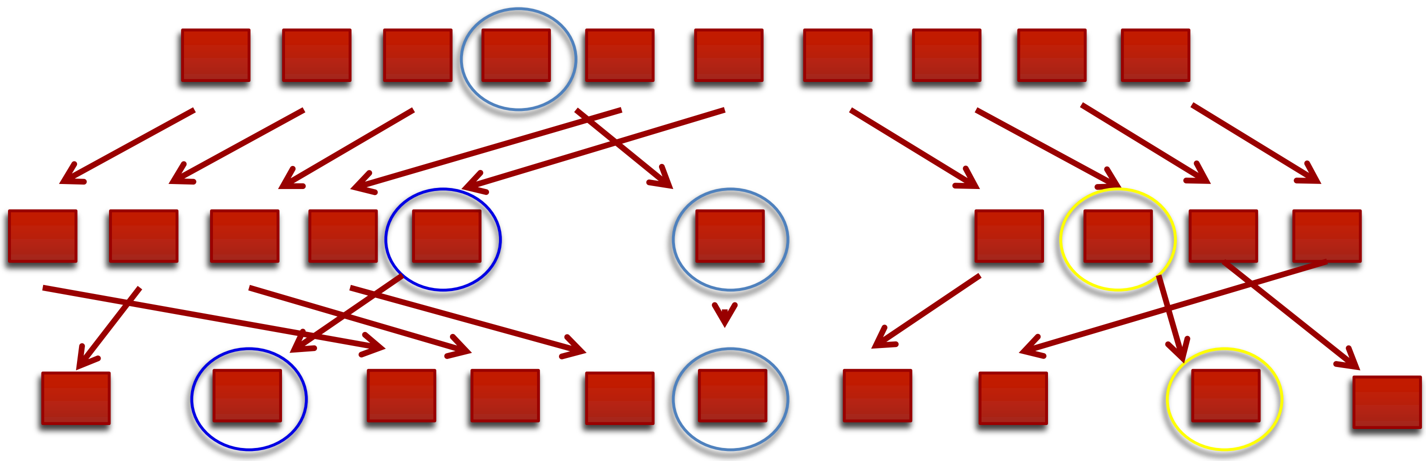


Quicksort

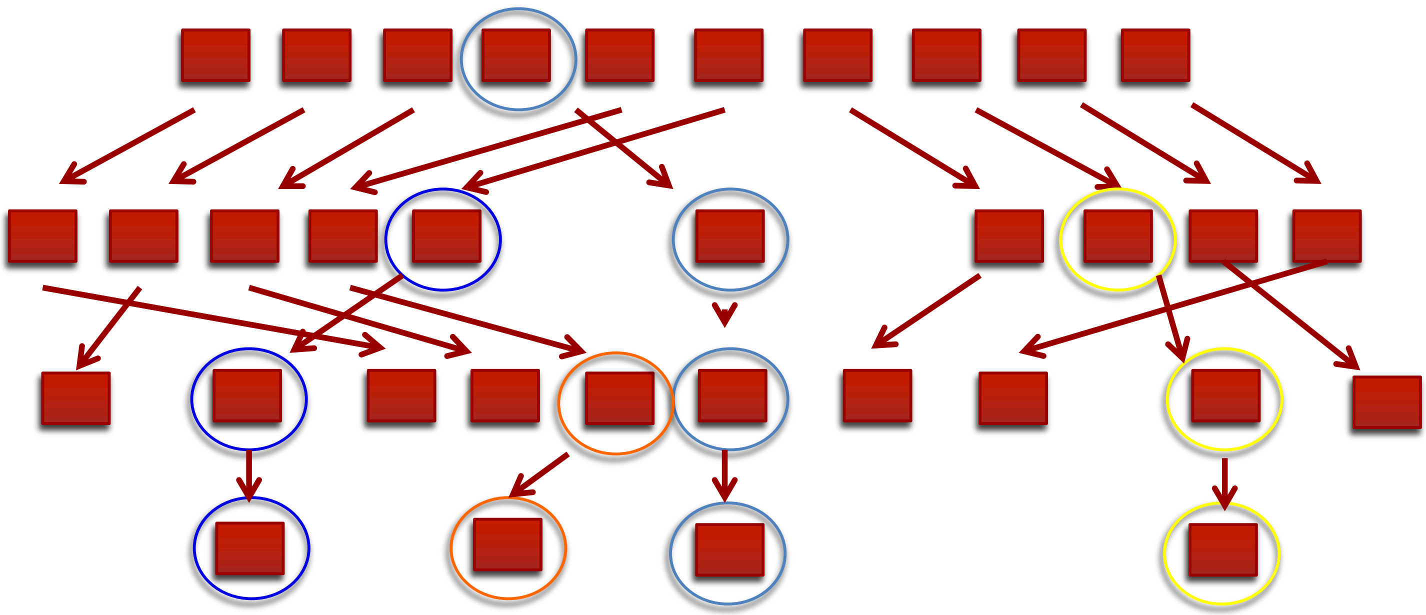


Left ‘list’ (1) Right list (3) Left list (2) Right ‘list’ (1)

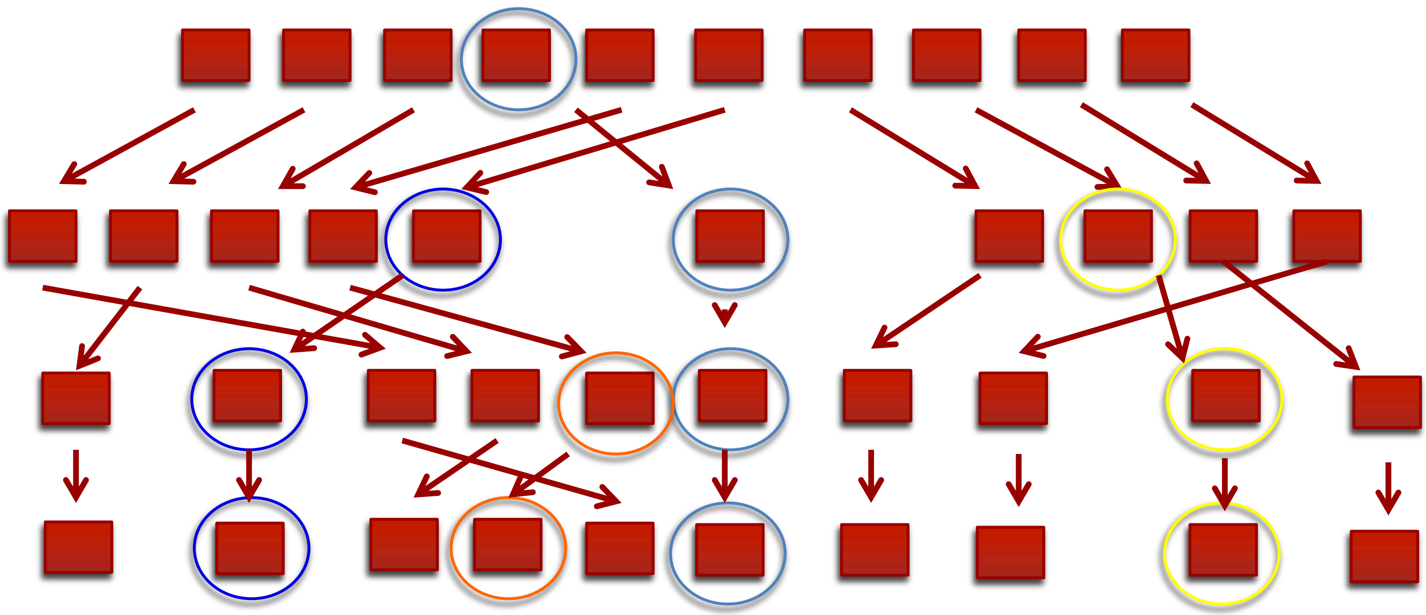
Quicksort



Quicksort



Quicksort



Sorting: Quicksort

13 4 3 5 12 6 20 10

Pivot = 6

4 3 5   6   13 12 20 10

Quicksort (4 3 5)

Quicksort (13 12 20 10)

Sorting: Quicksort

13 4 3 5 12 6 20 10

Pivot = 6

4 3 5	6	13 12 20 10
Quicksort (4 3 5)	6	Quicksort (13 12 20 10)
Pivot = 4		Pivot = 12
3 4 5	6	10 12 13 20
Quicksort(3) 4 Quicksort(5)	6	Quicksort(10) 12 Quicksort(13 20)
3 4 5	6	10 12 13 20

Sorting: Quicksort

13 4 3 5 12 6 20 10

Pivot = 12

4 3 5 6 10 12 13 20

Quicksort (4 3 5 6 10) 12 Quicksort (13 20)

Pivot = 4

3	4	5 6 10			12	13 20
Quicksort(3)	4	Quicksort(5 6 10)			12	13 20
3	4	Pivot = 6			12	13 20
3	4	Quicksort(5)	6	Quicksort(10)	12	13 20
3	4	5	6	10	12	13 20

Sorting: Quicksort

- If list is size 1, return the list
- If list is size 2, and out of order, swap elements and return the swapped elements, else return the list
- Pick an element in the sequence (called the pivot)
- Put all elements smaller than the pivot on its left
- Put all elements larger than the pivot on the right
- Sort the sequence on the left (use Quicksort)
- Sort the sequence on the right (use Quicksort)

Summary

- Solving a problem with a computer usually involves:
  - A structured way to store (organize) data
  - An algorithm that accesses and modifies that data
- Algorithms have characteristics, like *brute-force* or *divide-and-conquer* that help us understand how they work
- Thinking about abstract data types and algorithms frees us from worrying about the implementation details
- Sequences are a fundamental ADT used to organize data in an ordered list.

Homework Hints

- Problem #1
  - Work from inside-out, simplifying given the identities

- Problem #5
  - Don’t over think things, think of it like giving directions (e.g to the store)
  - Don’t need encodings, just use numbers
  - You do need to make loops
    - What happens if we write
      M0: SET R1 1
      M1: LOAD M104 R2
      M2: COND_GOTO R1 R2 M1
      …