

Introduction to Computer Science

CSCI 109

Readings

St. Amant, Ch. 4

Andrew Goodney

Spring 2017

"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

- ◆ HW1 due today
- ◆ Quiz 1 is next week (covers lecture material from 8/21 and 8/28)
- ◆ All quizzes will be held in class! Please review the quiz policy on the syllabus (or don't skip any quizzes... that's easy!)

Where are we?





Date	Topic		Assigned	Due	Quizzes/Midterm/Final
21-Aug	Introduction	What is computing, how did computers come to be?			
28-Aug	Computer architecture	How is a modern computer built? Basic architecture and assembly	HW1		
4-Sep	Labor day				
11-Sep	Data structures	Why organize data? Basic structures for organizing data		HW1	
12-Sep	Last day to drop a Monday-only class without a mark of "W" and receive a refund or change to Pass/No Pass or Audit for Session 001				
18-Sep	Data structures	Trees, Graphs and Traversals	HW2		Quiz 1 on material taught in class 8/21-8/28
25-Sep	More Algorithms/Data Structures	Recursion and run-time			
2-Oct	Complexity and combinatorics	How "long" does it take to run an algorithm.		HW2	Quiz 2 on material taught in class 9/11-9/25
6-Oct	Last day to drop a course without a mark of "W" on the transcript				
9-Oct	Algorithms and programming	(Somewhat) More complicated algorithms and simple programming constructs			Quiz 3 on material taught in class 10/2
16-Oct	Operating systems	What is an OS? Why do you need one?	HW3		Quiz 4 on material taught in class 10/9
23-Oct	Midterm	Midterm			Midterm on all material taught so far.
30-Oct	Computer networks	How are networks organized? How is the Internet organized?		HW3	
6-Nov	Artificial intelligence	What is AI? Search, planning and a quick introduction to machine learning			Quiz 5 on material taught in class 10/30
10-Nov	Last day to drop a class with a mark of "W" for Session 001				
13-Nov	The limits of computation	What can (and can't) be computed?	HW4		Quiz 6 on material taught in class 11/6
20-Nov	Robotics	Robotics: background and modern systems (e.g., self-driving cars)			Quiz 7 on material taught in class 11/13
27-Nov	Summary, recap, review	Summary, recap, review for final		HW4	Quiz 8 on material taught in class 11/20
8-Dec	Final exam 11 am - 1 pm in SAL 101				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

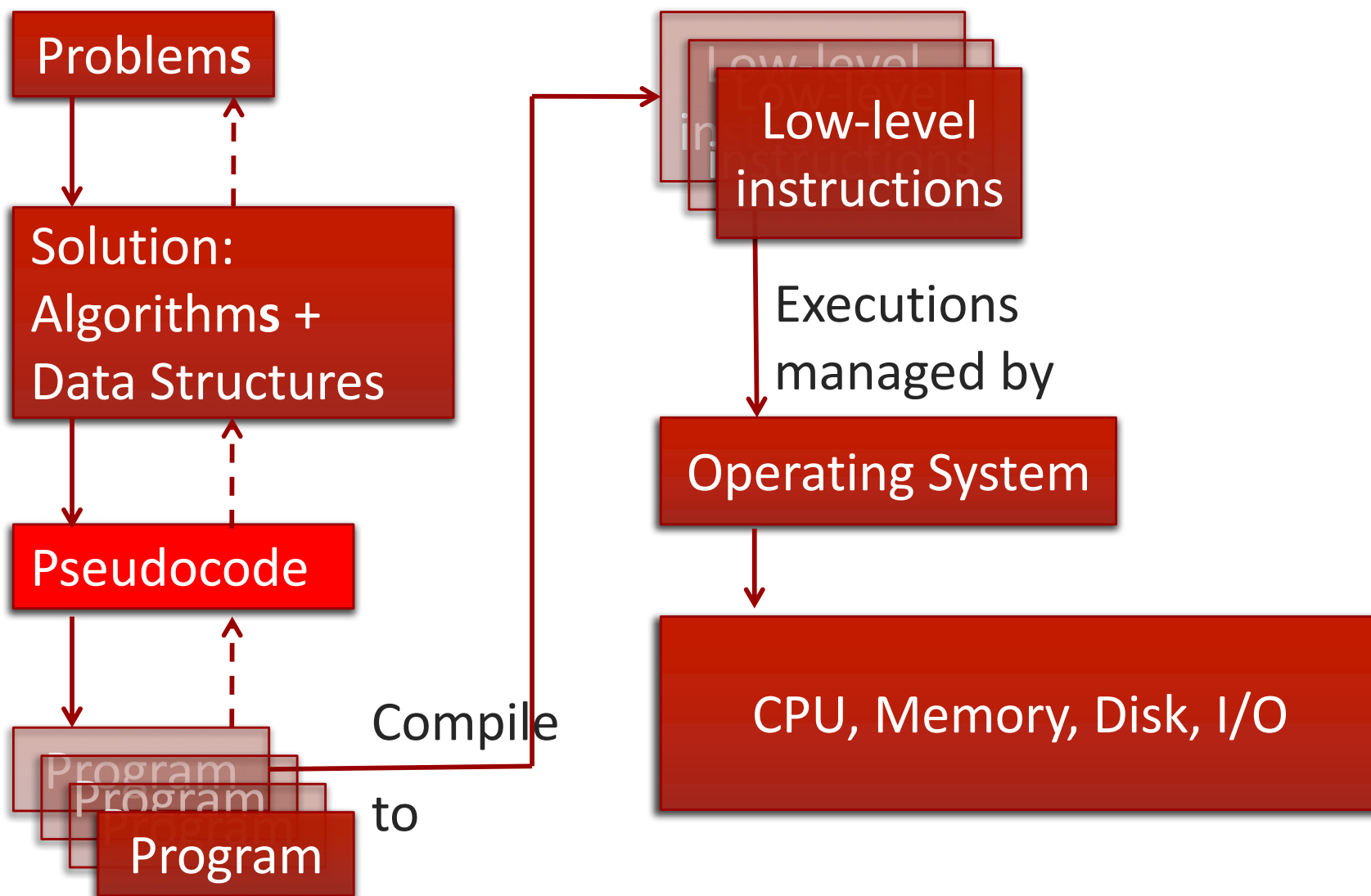
- 
- ◆ “The architecture level gives us a very detailed view of what happens on a computer. But trying to understand everything a computer does at this level would be...(insert analogy about perspective). If all we can see is fine detail, it can be hard to grasp what’s happening on a larger scale.”

- 
- ◆ “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

Overview



Problem Solving

- ◆ Architecture puts the computer under the microscope
- ◆ 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
 - ❖ Algorithms
 - ❖ Abstract data types

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

“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
- ◆ Typically at an abstract level
- ◆ Analogy: spices in your kitchen
 - ❖ Or pancake mix, or “bread crumbs”

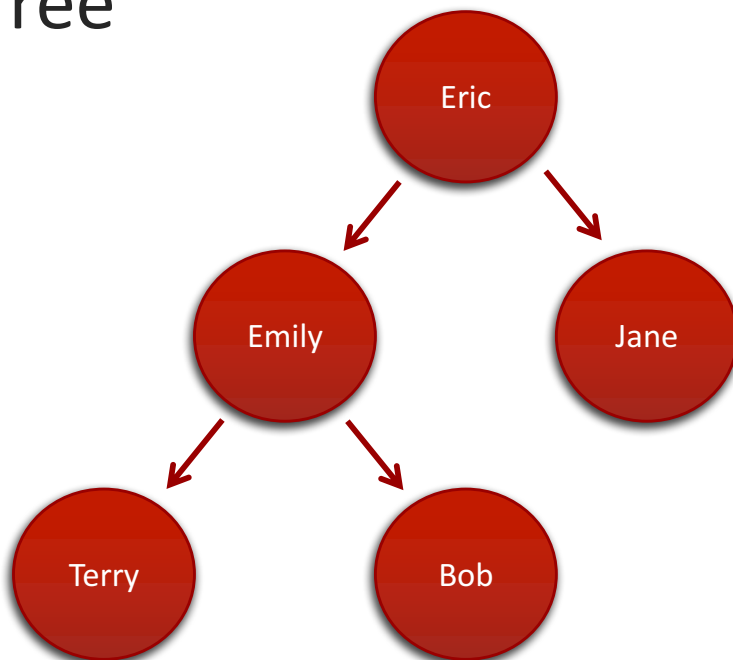
“... describes what can be done with a collection of information, without going down to the level of computer storage.” [St. Amant, pp. 53]

Sequences, Trees and Graphs

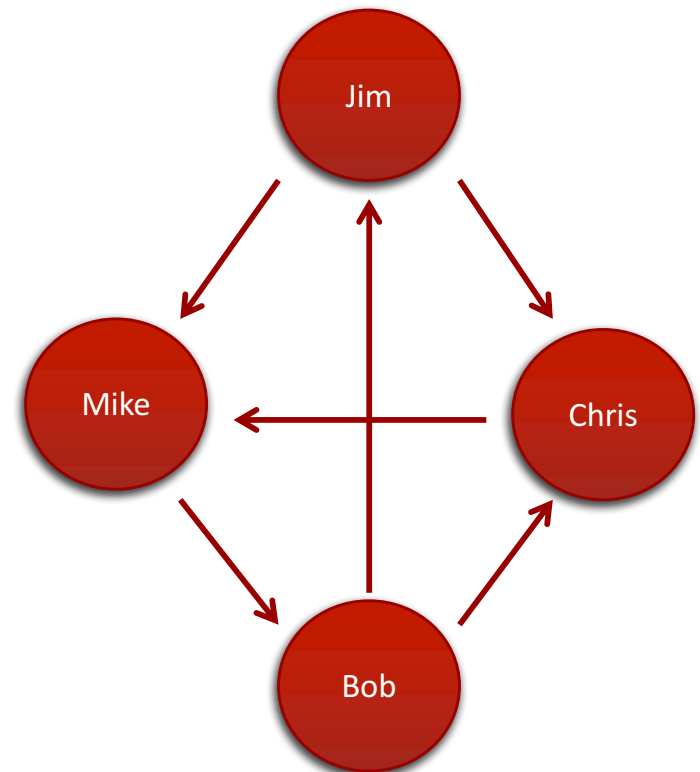
◆ Sequence: a list

- ❖ Items are called elements
- ❖ Item number is called the index

◆ Tree



◆ Graph



Is 'Jelena' on this list?

- | | | | |
|---------------|------------|-----------|-------------|
| ◆ Byron | ◆ Lenora | ◆ Jayna | ◆ Jayna |
| ◆ Therese | ◆ Staci | ◆ Joy | ◆ Jesusa |
| ◆ Alpha | ◆ Ena | ◆ Sean | ◆ Dion |
| ◆ Christopher | ◆ Elsy | ◆ Basilia | ◆ Orpha |
| ◆ Jacquelyn | ◆ Derrick | ◆ Classie | ◆ Denice |
| ◆ Amada | ◆ Kelley | ◆ Sharice | ◆ Tad |
| ◆ Araceli | ◆ Kathe | ◆ Carina | ◆ Geraldine |
| ◆ Deanna | ◆ Mohammad | ◆ Livia | ◆ Bradley |
| ◆ Mario | ◆ Julia | ◆ Klara | ◆ Mariah |
| ◆ Pamela | ◆ Renda | ◆ Bess | ◆ Lyndsey |
| ◆ Lin | ◆ Kylee | ◆ Simone | ◆ Marcia |
| ◆ Hester | ◆ Keren | ◆ Michiko | ◆ Beatrice |
| | | ◆ Elmer | ◆ Keri |
| | | | ◆ Thu |

Is 'Lilly' on this list?

- | | | | | |
|--------------|-------------|-----------|-------------|-----------|
| ◆ Allene | ◆ Exie | ◆ Jenae | ◆ Machel | ◆ Rosann |
| ◆ Berenice | ◆ Ezequiel | ◆ Joanne | ◆ Madelaine | ◆ Sally |
| ◆ Bernadine | ◆ Filiberto | ◆ Jonie | ◆ Marielle | ◆ Season |
| ◆ Candelaria | ◆ Francisca | ◆ Lannie | ◆ Mauro | ◆ Sidney |
| ◆ Carli | ◆ Fred | ◆ Leanora | ◆ Mayola | ◆ Tamica |
| ◆ Carry | ◆ Gayle | ◆ Lilliam | ◆ Mikaela | ◆ Tilda |
| ◆ Chau | ◆ Gudrun | ◆ Lilly | ◆ Pamala | ◆ Val |
| ◆ Cinthia | ◆ Huey | ◆ Lina | ◆ Pinkie | ◆ Vinita |
| ◆ Clement | ◆ Isaiah | ◆ Lorinda | ◆ Princess | ◆ Yaeko |
| ◆ Davina | ◆ Janey | ◆ Lulu | ◆ Rocco | ◆ Yoshiko |

Sequences: Searching

- ◆ Sequential search: start at 1, proceed to next location... *brute force*
- ◆ 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
 - ❖ ...*divide-and-conquer*
- ◆ Which is more efficient?

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

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

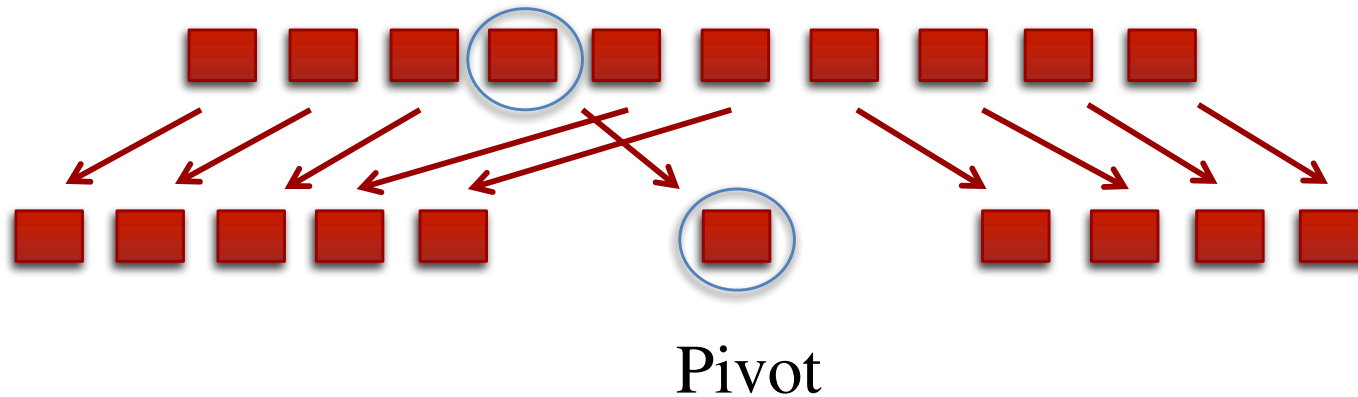
- ◆ Pick a 'middle' element in the sequence (this is called the pivot)
- ◆ Put all elements smaller than the pivot on its left
- ◆ Put all elements larger than the pivot on the right
- ◆ Now you have two smaller sorting problems because you have an unsorted list to the left of the pivot and an unsorted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



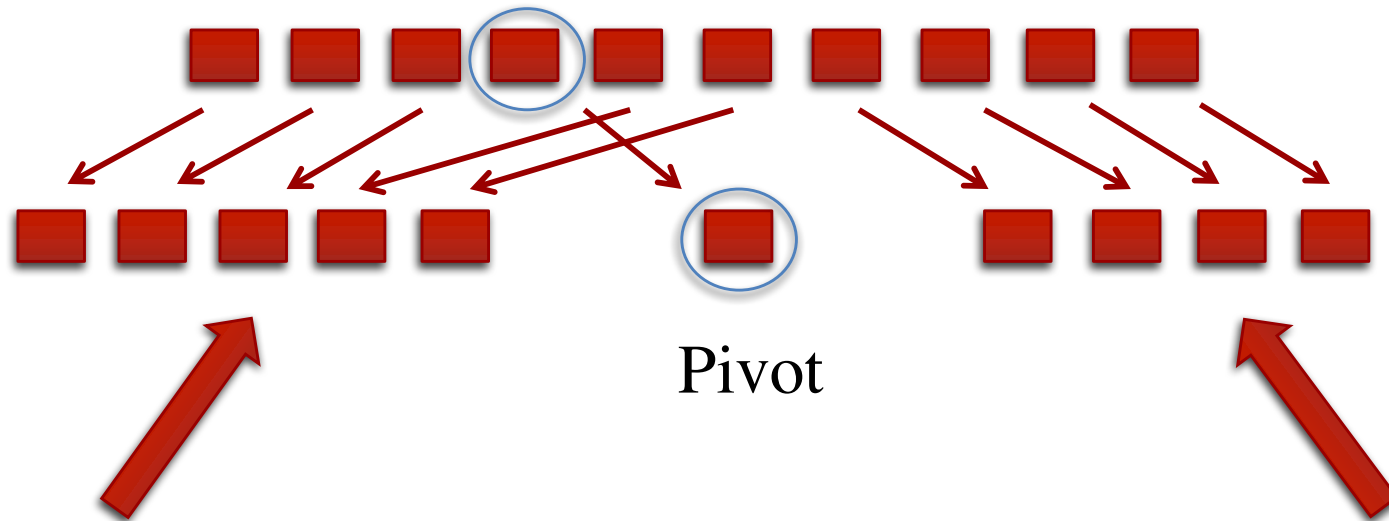
This is an unsorted list (e.g.,
a list of numbers not in
order)

Quicksort



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

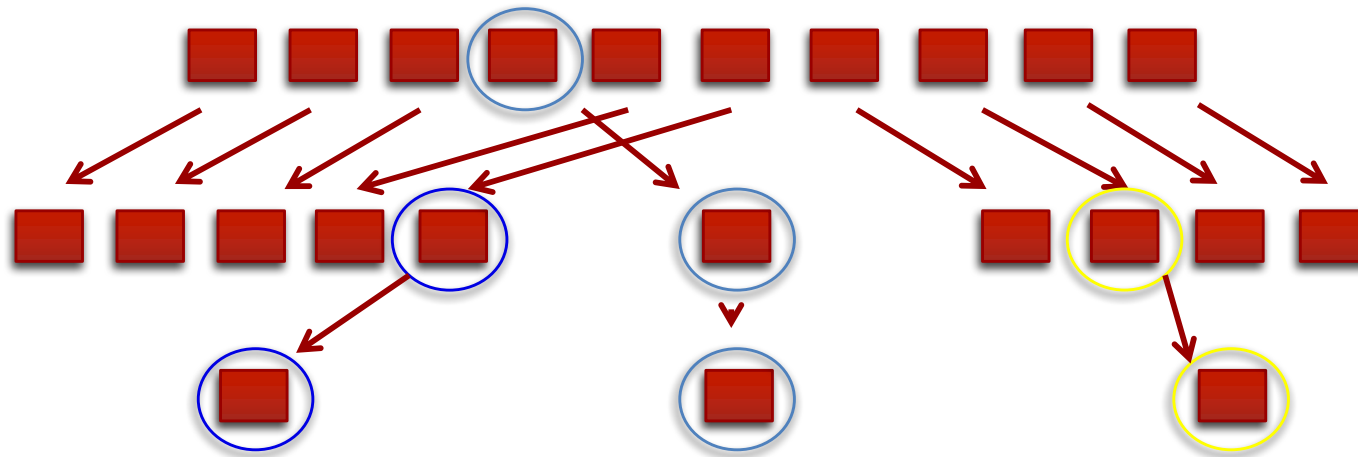
Quicksort



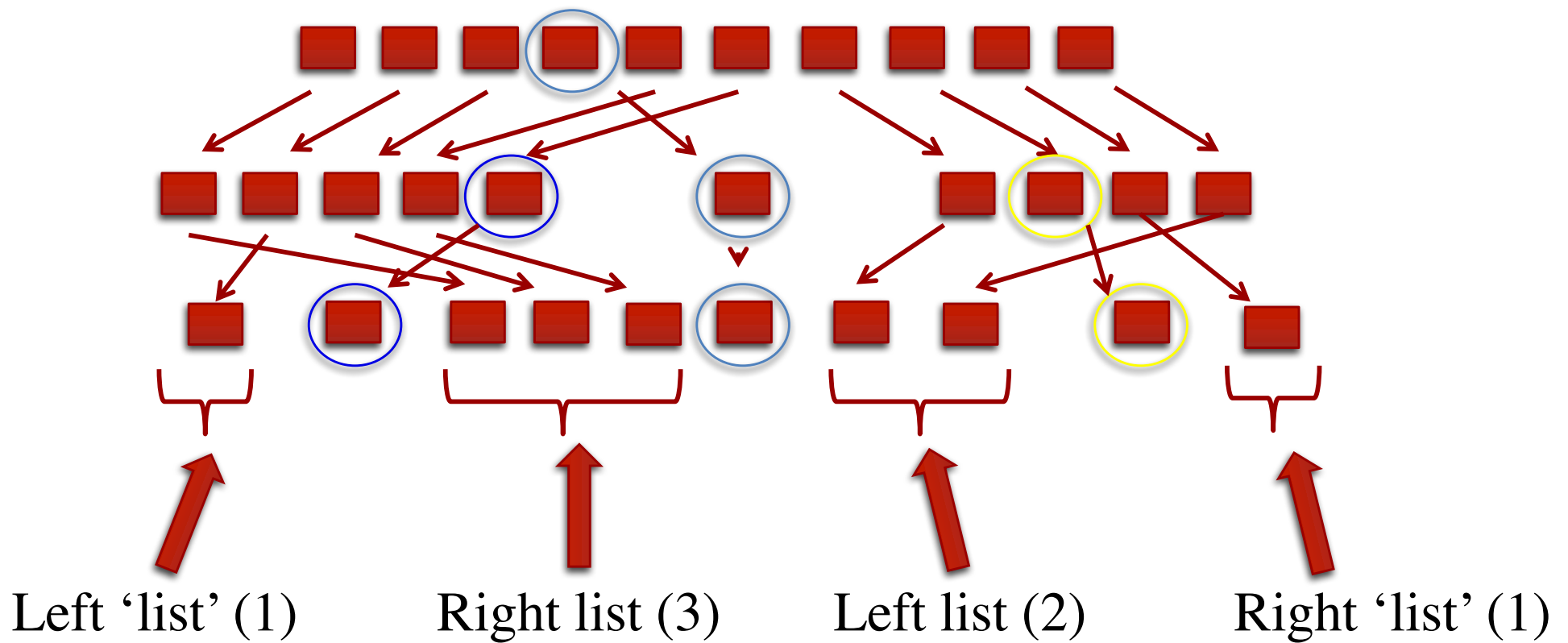
This is an unsorted list of all elements smaller than the pivot

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

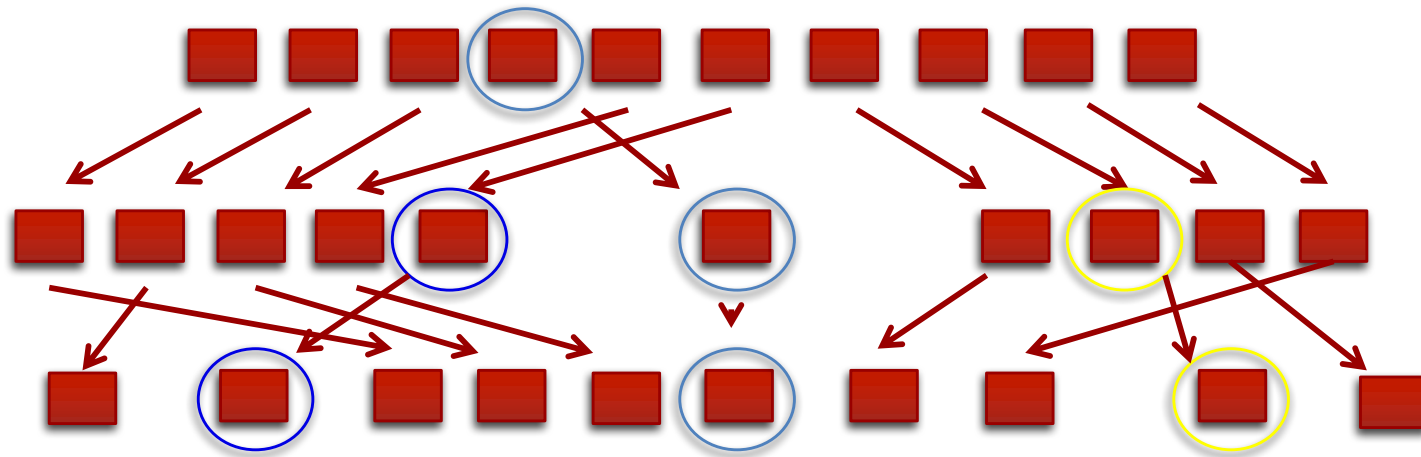
Quicksort



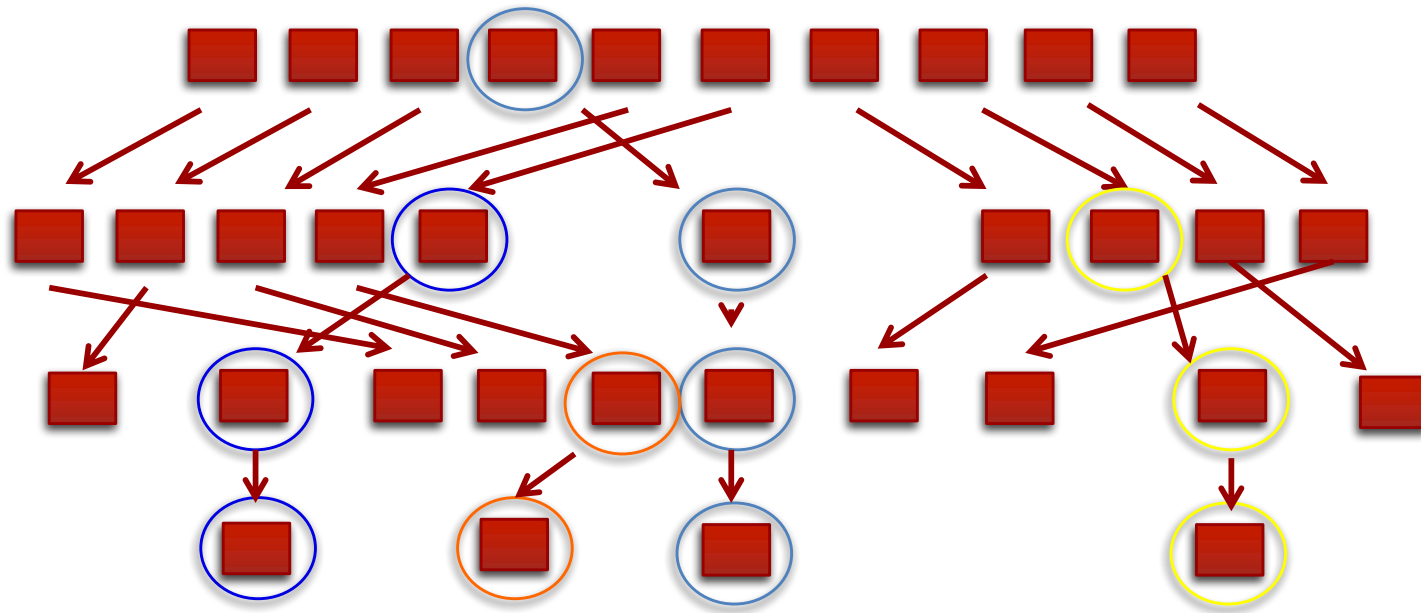
Quicksort



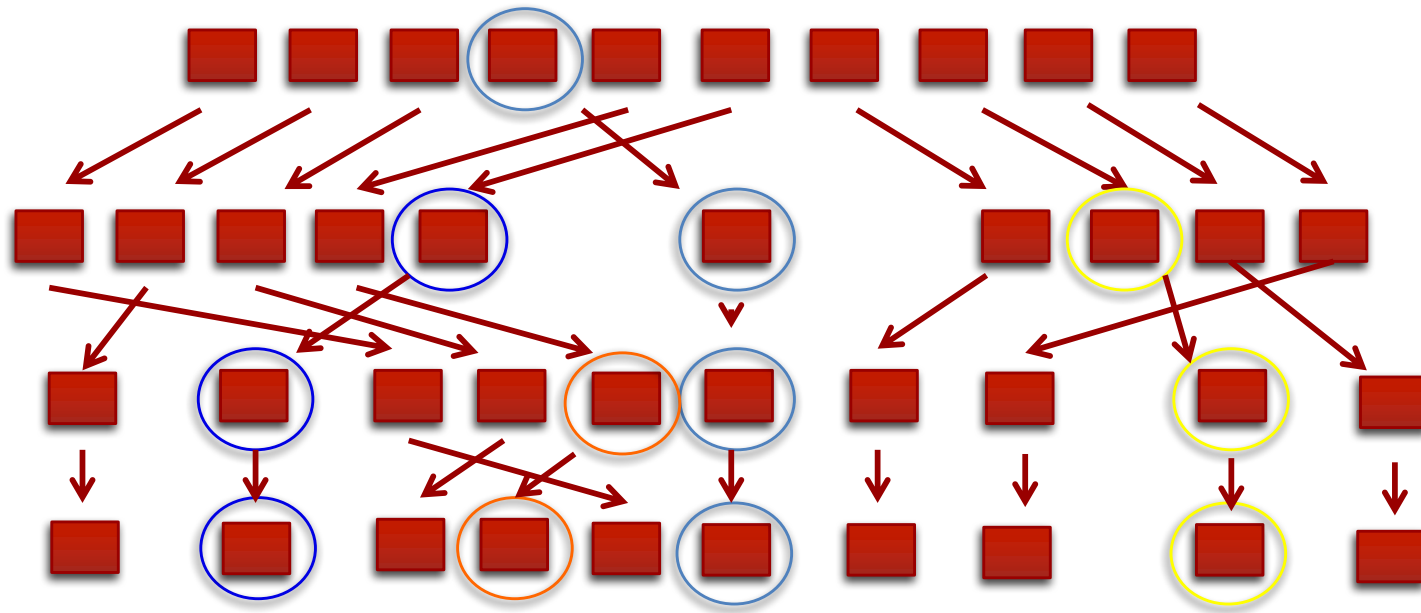
Quicksort



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

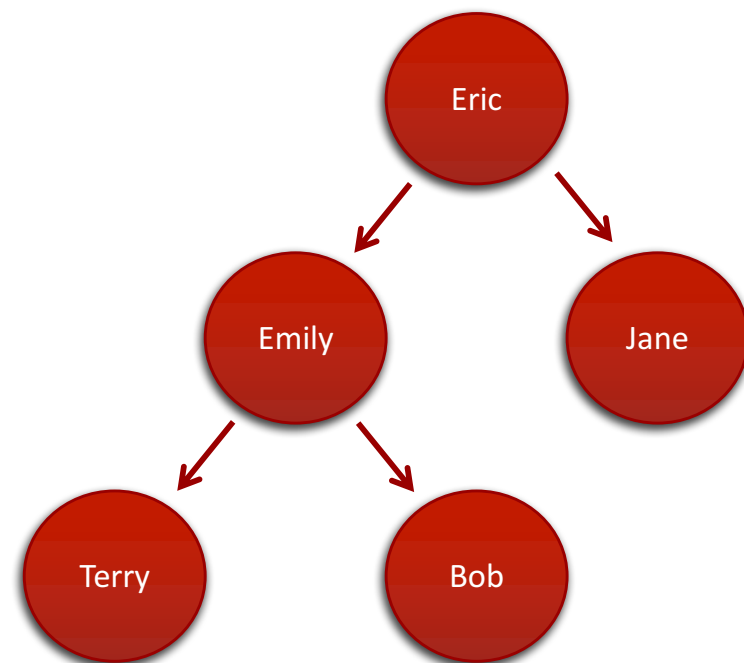
13 20

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)

Trees

- ◆ Each node/vertex has exactly one parent node/vertex
- ◆ No loops
- ◆ Directed (links/edges point in a particular direction)
- ◆ Undirected (links/edges don't have a direction)
- ◆ Weighted (links/edges have weights)
- ◆ Unweighted (links/edges don't have weights)



Which of these are NOT trees?

