CSCI 104 Overview

Sandra Batista

1.1–1.2

Welcome to CSCI 104!

Hope you are all safe, healthy and doing well!

I want to meet you! I created an introductory survey if you want to introduce yourself to me:

https://piazza.com/class/kdfb05j88b15pb?cid=7

For now, let's meet each other! Take a moment to meet a few classmates. Please share some of the following answers:

- What is your name, major, and year?
- What is one thing about you your classmates may be surprised to learn that has nothing to do with CSCI 104 or computer science?

Administrative Details

Preparation for CSCI 103 (or equivalent course)

- Basic if, while, for constructs
- Arrays, linked-lists
- Structs, classes
- **Dynamic memory allocation and pointers**
- **Recursion**

Course Website:

- https://bytes.usc.edu/cs104/

Let’s take a moment to review syllabus and the course resources on the website

Office Hours

Office Hours Start Week 2 for entire course staff

Please make sure you followed the instructions in Lab 0 to access the Office Hours Queue.

My Office Hours: Fridays 9- 11 am, Thursdays 3:30-5:30 pm

My Social Tea Hour: Wednesdays 2-3 pm

This Friday I will have more of a meet and greet office hours. Please feel free to stop by. We can just socialize or you can ask technical questions

https://piazza.com/class/kdfb05j88b15pb?cid=15

Real-Time Meeting Accommodations

We want you to know we are sympathetic to your needs and want to support you. If you require an accommodation for real-time meetings such as labs, lectures, or office hours, please fill out this form:

https://piazza.com/class/kdfb05j88b15pb?cid=17

We will offer you the best accommodation we can manage.

Problem Solving: Are these circles connected?

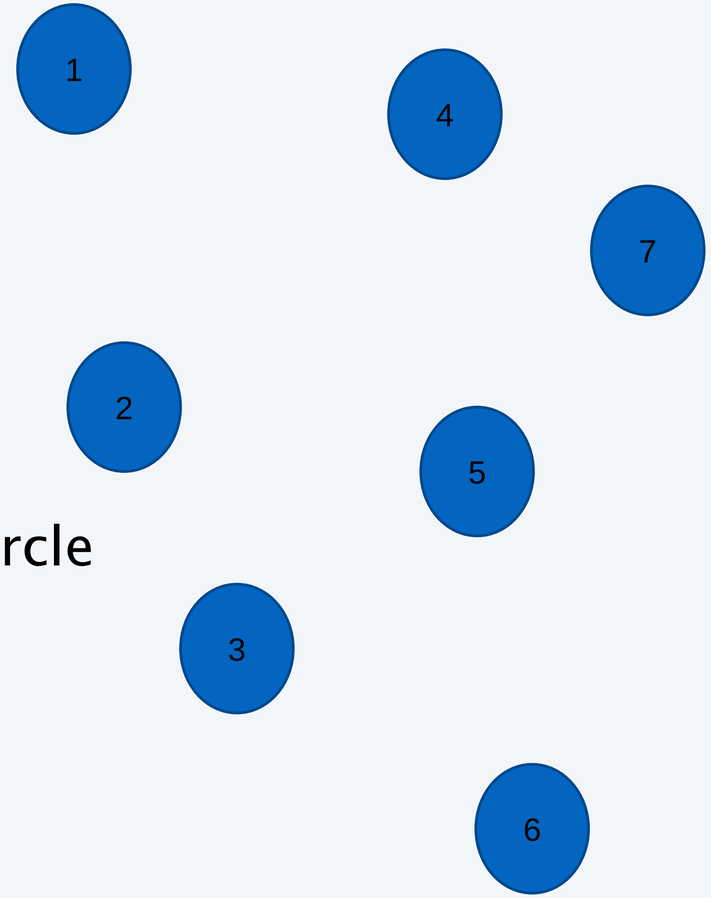
Suppose we have these 7 circles. If we want to connect two circles, we use the command connect(x,y)

connect(1,2)

connect(1,4)

connect(5,6)

connect(7,5)



We can store whatever we want about those connect calls in an int array of size 7, one for each circle

1	2	3	4	5	6	7

Are these circles connected?

Now we have a query path(x,y):

Are there lines connecting circle x and y?

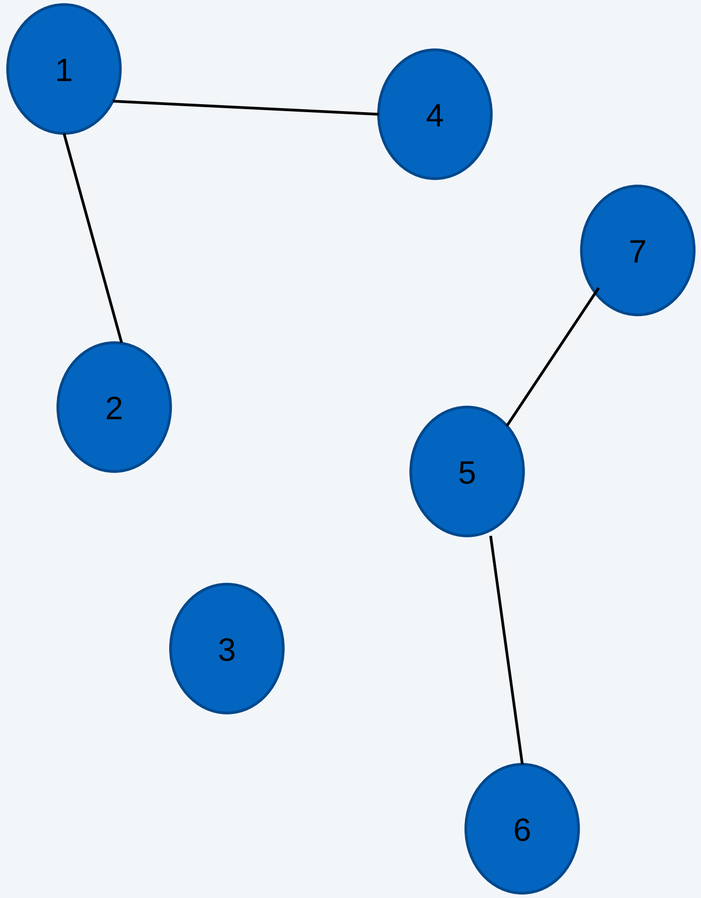
Path(1, 2) yes!

Path(2, 4) yes!

Path(4, 7) no

How can we use our array to help us?

1	2	3	4	5	6	7



Are these circles connected?

For each connect (x,y) call:

change all values with array[y-1] to value array[x-1]

Starting:

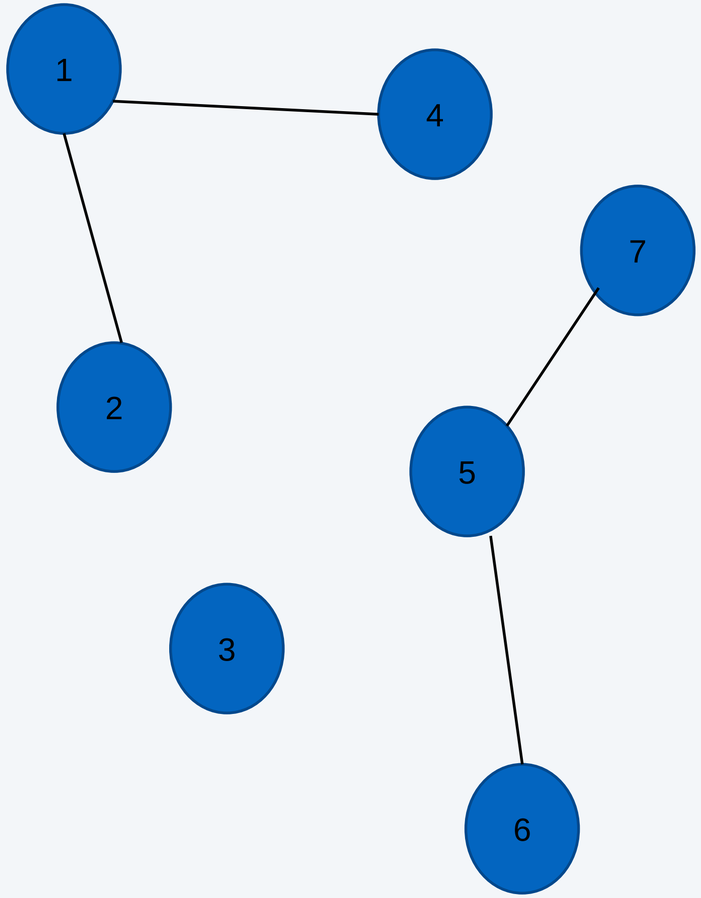
1	2	3	4	5	6	7

After connect(1,2)

1	1	3	4	5	6	7

After connect(1,4)

1	1	3	1	5	6	7



Are these circles connected?

For each connect (x,y) call

change all values with array[y-1] to value array[x-1]

After connect(1,4):

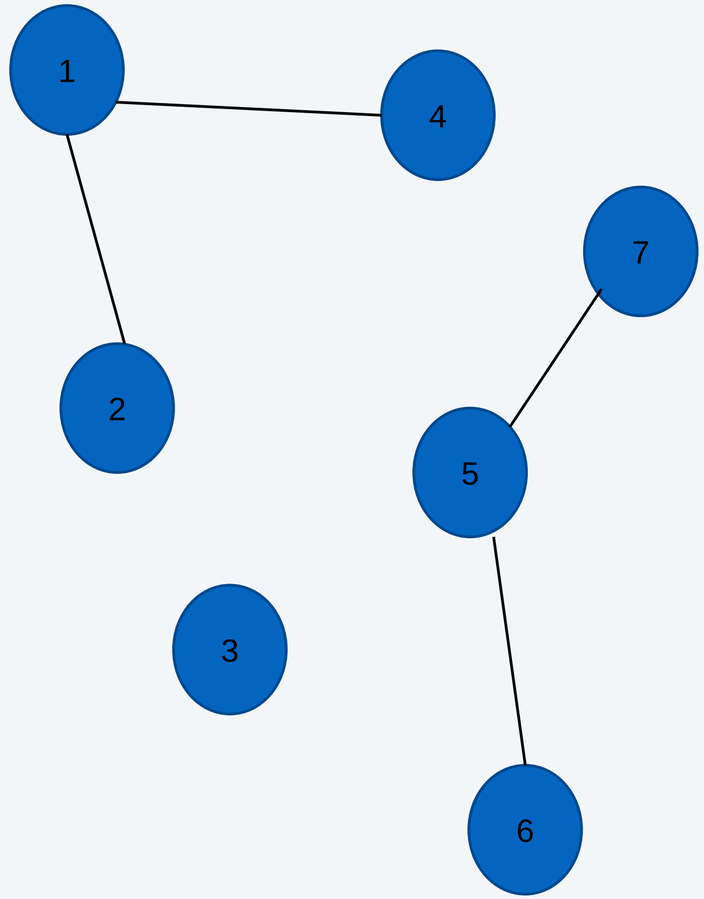
1	1	3	1	5	6	7

After connect(5,6)

1	1	3	1	5	5	7

After connect(7,5)

1	1	3	1	7	7	7



Are these circles connected?

For queries path(x,y):

Are there lines connecting circle x and y?

Path(1, 2) yes!

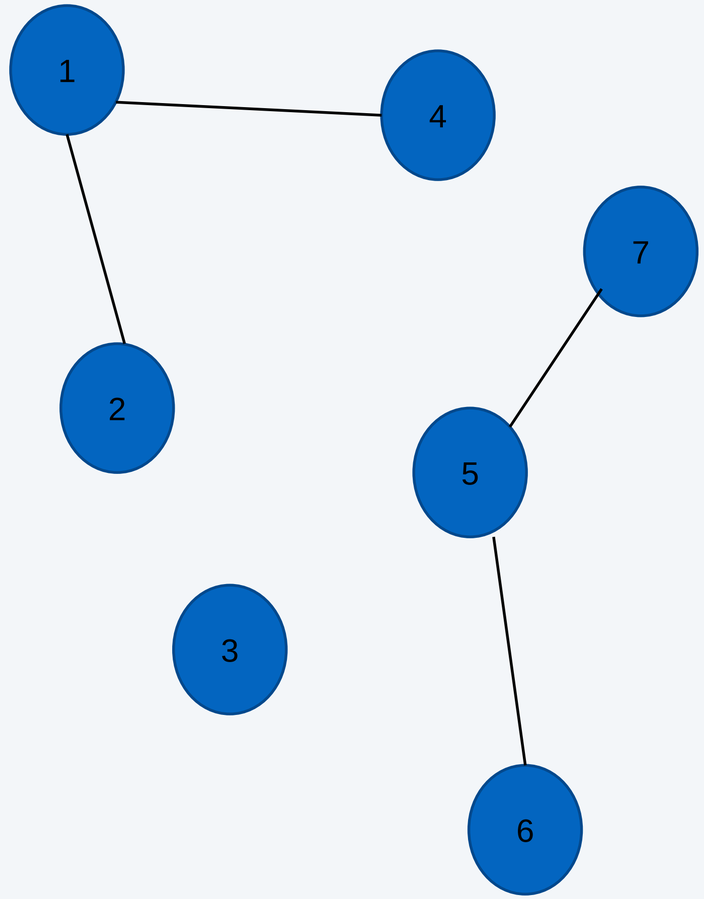
Path(2, 4) yes!

Path(4, 7) no

To answer path(x,y):

check if array[x-1] == array[y-1]

1	1	3	1	7	7	7



Problem Solving: Circles Connected Example

How we used our data, our array, was influenced by the problem we wanted to solve.

How we used our data, our array, influenced how efficient our solution is.

Key Points:

1. **The best way to organize data depends on how it will be used.**
2. **How we organize and use data affects the efficiency of our solution.**

Another Motivating Example

- Creating a USC Social Activities Search (USC SAS)

– Stores names of people, their phone numbers, their email addresses, their majors, their hobbies

- What operations do we perform with this data
  - You: Lookup/search
  - Admin: Add, Remove
- How is the data stored and ordered and why?

– Sorted by ???

– How fast?

Would it ever be reasonable to have SAS in unsorted order? What should we consider in ordering?

Key points:

1. The best way to organize data depends on how it will be used.

- Frequent search
- Frequent addition/removals
- Addition/removal patterns (many at once or one at a time)

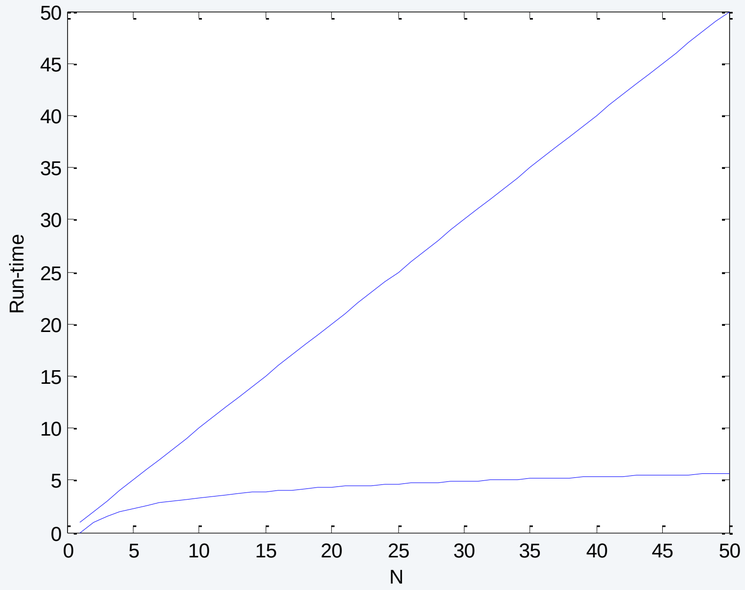
2. How we organize data we use affects the efficiency of our solutions

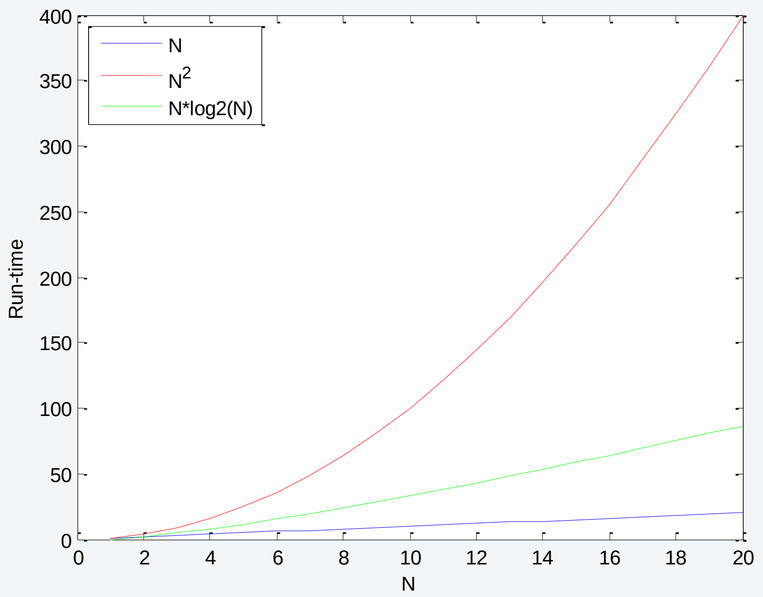
Why Data Structures Matter?

- Modern applications process vast amount of data
- Adding, removing, searching, and accessing are common operations
- Various data structures allow these operations to be completed with different time and storage requirements

Data Structure	Insert	Search	Get-Min
Unsorted List	O(1)	O(n)	O(n)
Balanced Binary Search Tree	O(lg n)	O(lg n)	O(lg n)
Heap	O(lg n)	O(n)	O(1)

Importance of Complexity





N	O(1)	O(log n)	O(n)	O(n log n)	O(n²)	O(2ⁿ)
2	1	1	2	2	4	4
20	1	4.3	20	86.4	400	1,048,576
200	1	7.6	200	1,528.8	40,000	1.60694E+60
2000	1	11.0	2000	21,931.6	4,000,000	#NUM!

Object-oriented paradigm

- Focus on data stored and operations performed on that data

C++ Classes allows programmer to group data and operations into a logical unit

C++ Classes provide abstraction and encapsulation.

Abstraction: Hide implementation details behind a simplified interface

E.g. - use <algorithm> std::sort without knowing how it is implemented

- offer well designed public functions for classes

Encapsulation: Protects implementation details of the data and operations on the data from external direct use

E.g.- use private and protected member data and functions in classes

Learn good object-oriented design practices:

- Use of advanced C++ topics such as templates, inheritance, polymorphism, etc.
- Develop easy to read and maintain code: Good OO code should read like English

Learn basic and advanced techniques for implementing data structures and analyzing their efficiency

Improve Problem Solving Skills

Encourage Intellectual Curiosity