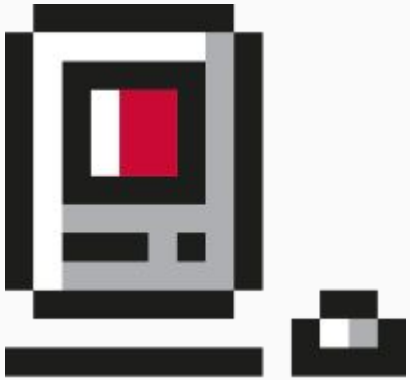
ACM CSIP Summer 2021

Meeting 5! Check-in pls: https://forms.gle/w3FK9PznnXuSWN8g9

Welcome!

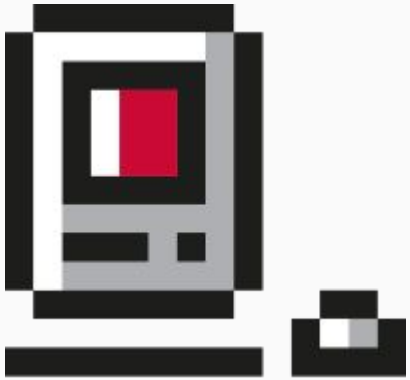
- Itinerary
  - Introduction
  - Polls
  - Big O
  - Time Complexity
  - Space Complexity
  - Trade Offs & Process
  - Resources



Introduction

Introduction

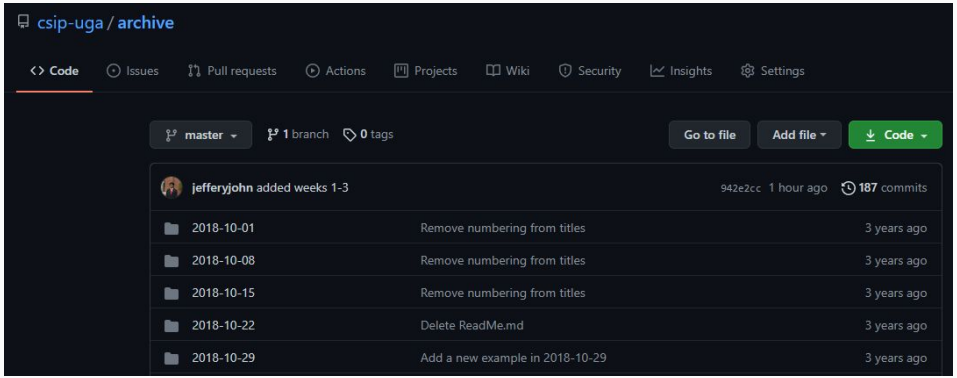
- Computer Science Interview Prep (CSIP)
- Meeting Tuesdays @ 7PM, ACM Discord
- Focused on professional, personal, and technical development
- Will be focused on both behavioral and technical interviews

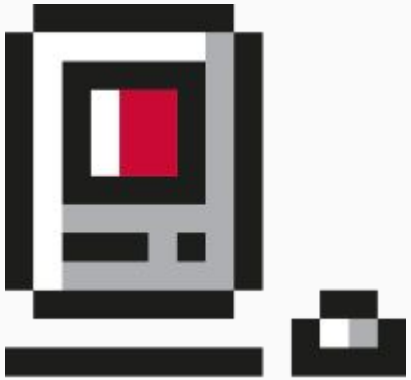


Meeting Platforms

GitHub

- github.com/csip-uga
- share your GitHub username to be added!

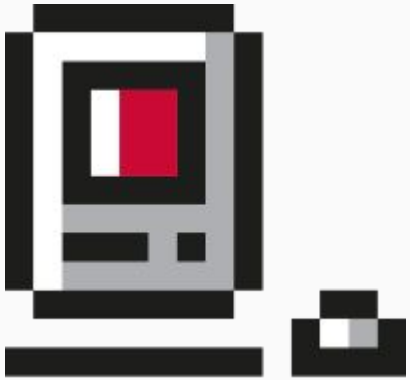




Polls

Polls

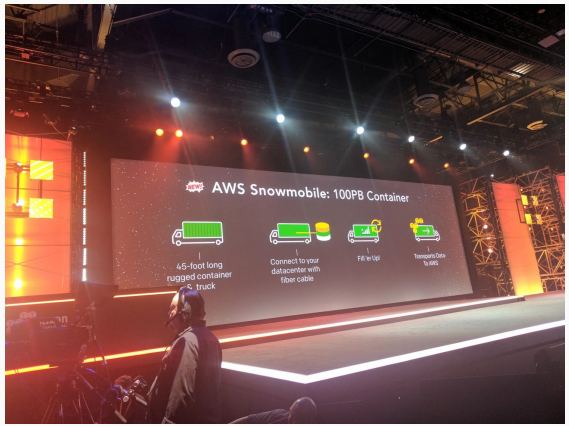
- See check-in form
- Have you taken CSCI 2720: Data Structures?
- Do you feel confident in analyzing time and space complexity?
- Do you feel confident in technical interviews?

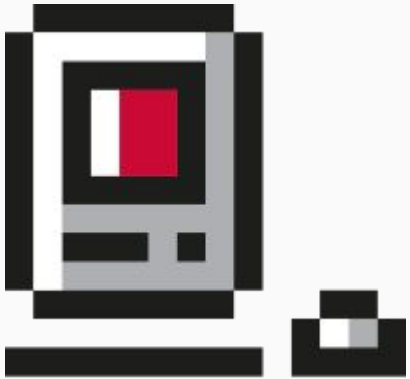


Big O

What is Big O?

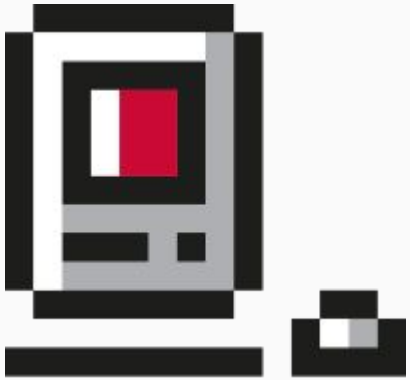
- Data is, like, really Big
  - AWS Snowmobile
- Refers to Time and Space Complexity



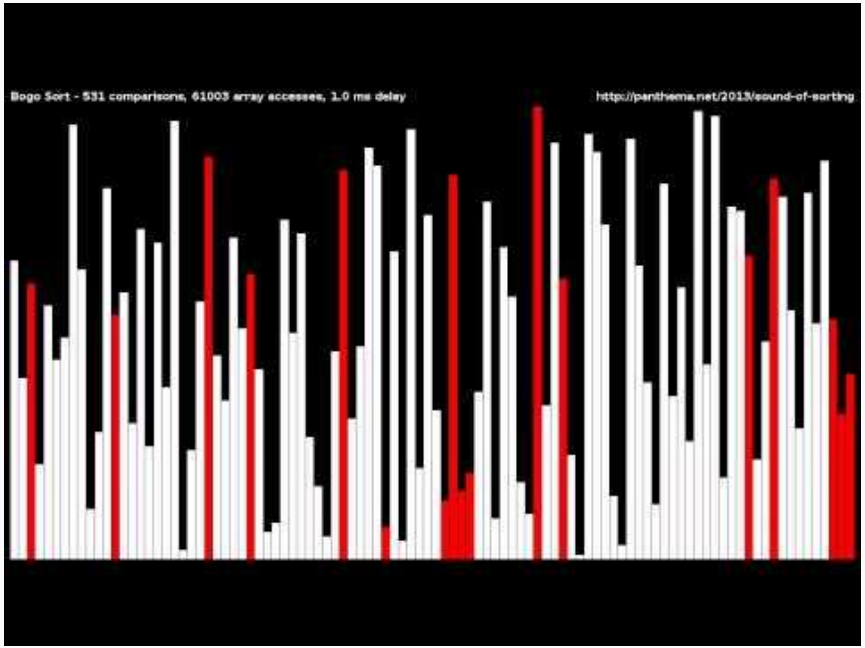


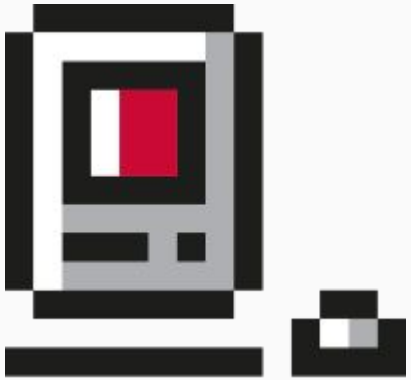
Academic Definitions

- Big O: describes an *upper* bound
- Big Omega (Ω): describes a *lower* bound
- Big Theta Θ: O *and* Ω
  - Use this for interviews
  - AKA give the best answer you can
  - Consider scale and dominant terms, not constants



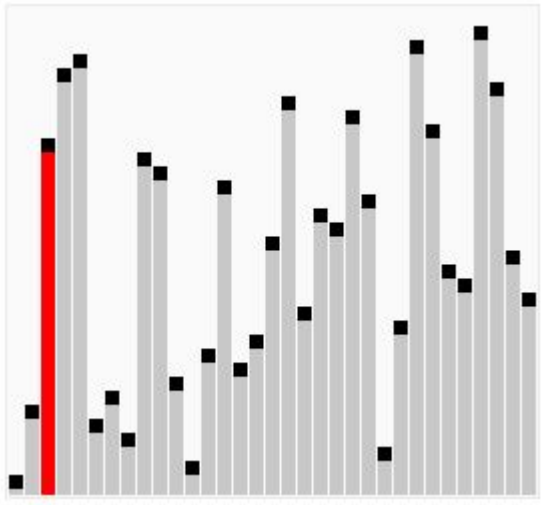
Best, Worst, and Expected

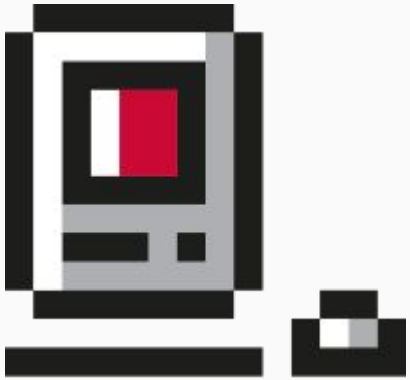




Best, Worst, and Expected

- Bogo Sort
  - Best: n
  - Worst: Infinite
  - Expected: n * n!
- Bubble Sort
  - Best: n
  - Worst: n^2
  - Expected: n^2
- Complexities
  - n!, 2^n, n^2, n * log n, n, log n, 1
  - Base of log is constant

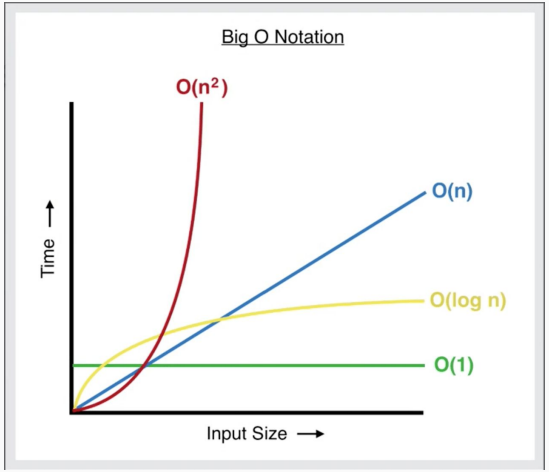


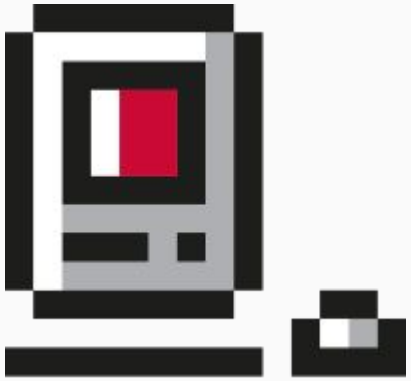


Time Complexity

Time Complexity

- Big O runtime - what most people care about
- O(s) - where s is the size of the file you have to upload to AWS
- O(1) - where the size of the file doesn't matter to the truck

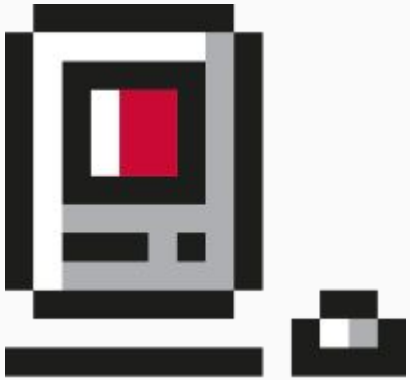




Space Complexity

Space Complexity

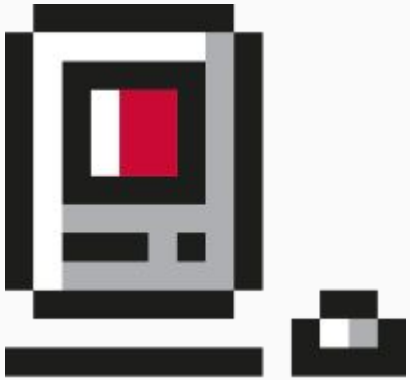
- It takes O(n) space to hold n elements
  - Array of size n
- Recursive calls can take space too
  - Factorial: O(n)
  - Pairing Sequence of Sums: O(1), even w/ n calls
  - Multiple calls: O(branches^depth)
- Search string S then string T
  - O(S + T) space



Extending Complexity

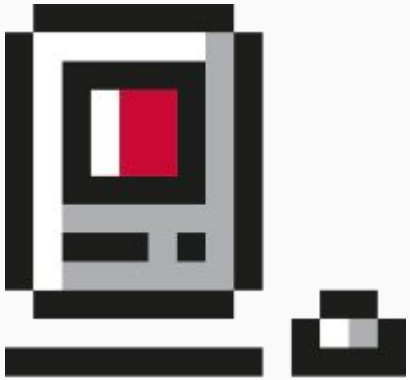
Combining Runtimes

- If A *then* B, O(A+B)
- If B *for each of* A, O(A * B)
- Think about how your loops interact with each other
- Consider that A and B may have different lengths, so they aren't equivalent or constants
    - The time to find a computer science classroom at UGA scales with distance to building D, numbers of floors F
    - The time to attend via Zoom is O(1)



Amortized Runtime

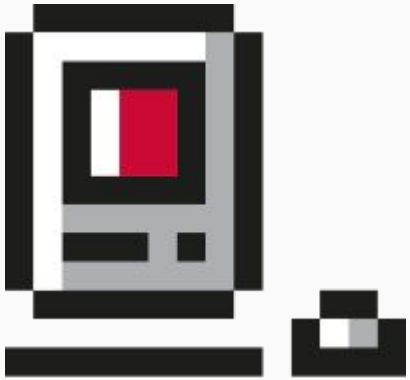
- Dynamic array: CSCI 1302, CSCI 1730, CSCI 2720
- When an array reaches full capacity, create another with double the elements
- Arrays have O(1) add time
    - array[i] = element
- Copying an array takes O(N) time
    - for (int i = 0; i < array.length; i++)
- Does adding an element always take O(1) or O(N) time?
- We double its size at 1, 2, 4, 8, 16, X
- 1 + 2 + 4 + 8 + 16 + … + X (adding powers of 2)
    - X + X / 2 + X / 4 + X / 8 + …. + 1 (dividing by powers of 2)
    - = 2X, or X adds take O(X) time
    - = O(X) / X = O(1) time for individual additions



Big O Examples

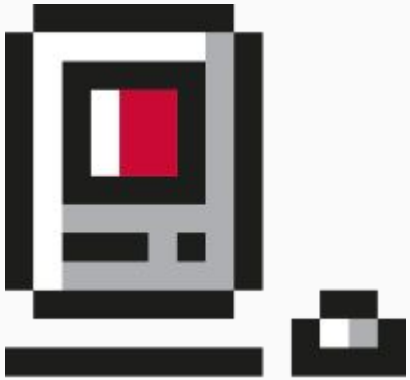
Example 1

```
for (int i = a.length - 1; i >= 0; i--) {
        System.out.println("Welcome to ACM-CSIP");
}
for (int i = 0; i < a.length; i++) {
        System.out.println("Join us on GitHub");
}
```



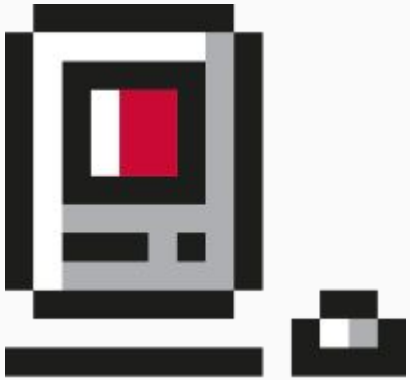
Example 2

```
for (int i = 0; i < a.length; i++) {
        System.out.println("Welcome to ACM-CSIP");
}
for (int i = 0; i < b.length; i++) {
        System.out.println("Join us on GitHub");
}
```



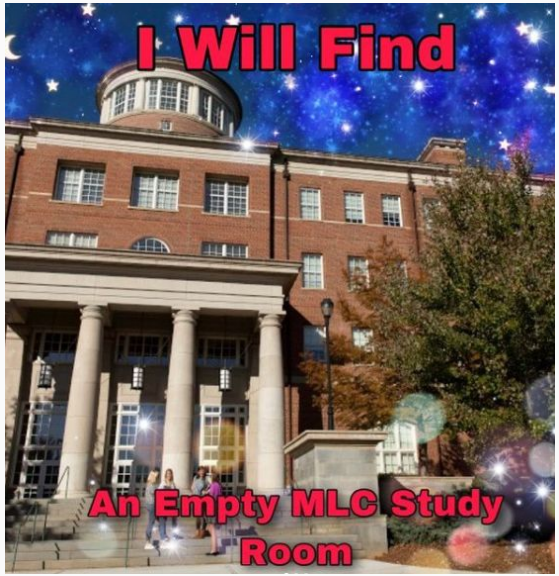
Example 3

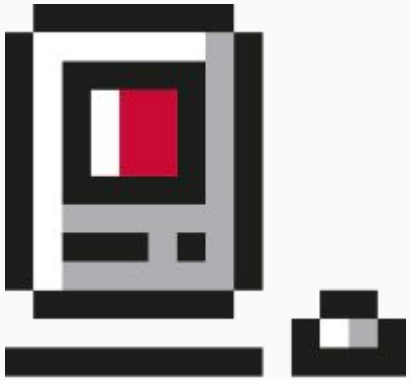
```
for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a.length; j++) {
                System.out.println(array[i] + array[j] + “ reasons to join”);
        }
}
```



Example 4

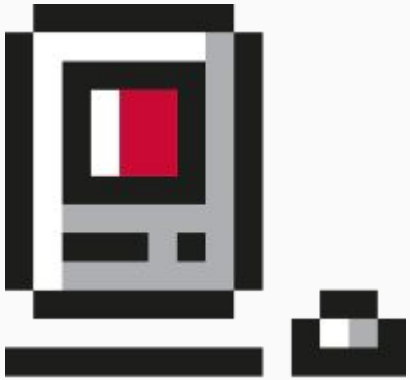
```
int waitlistSize = 1000

for (int k = 0; k < waitlistSize; k++) {

        System.out.println(“please”);

}
```





Example 5

```
int cutenessOfMeetingAttendees= 1000000
for (int i = 0; i < a.length / 2; i++) {
        for (int k = 0; k < cutenessOfMeetingAttendees; k++) {
                System.out.println(“<3”);
        }
}
```



Trade-Offs & Process

Data Structures

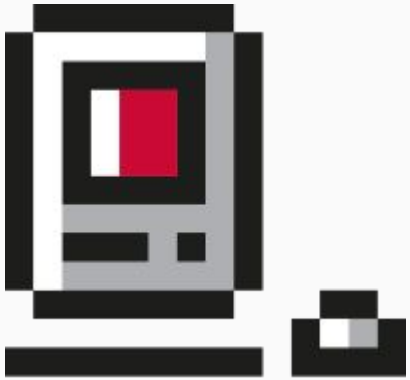
- Consider strengths and weaknesses of your implementation

Comparing the General-Purpose Storage Structures

Table 15.1 summarizes the speeds of the various general-purpose data storage structures using Big O notation.

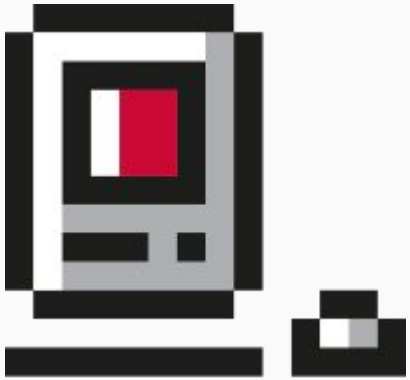
TABLE 15.1 General-Purpose Data Storage Structures

Data Structure	Search	Insertion	Deletion	Traversal
Array	O(N)	O(1)	O(N)	—
Ordered array	O(logN)	O(N)	O(N)	O(N)
Linked list	O(N)	O(1)	O(N)	—
Ordered linked list	O(N)	O(N)	O(N)	O(N)
Binary tree (average)	O(logN)	O(logN)	O(logN)	O(N)
Binary tree (worst case)	O(N)	O(N)	O(N)	O(N)
Balanced tree (average and worst case)	O(logN)	O(logN)	O(logN)	O(N)
Hash table	O(1)	O(1)	O(1)	—



Big O Influence

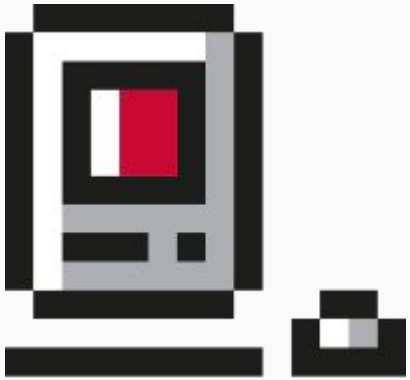
- Comparing two sorted arrays of same length
  - Iterate through both: O(2N) = O(N)
  - Brute force: O(N * N) = O(N^2)
  - Iterate + Binary Search: O(N * log N)
  - Iterate + Hash Table: O(N * 1) = O(N) *time*
  - Modified Linear: O(N) time *and* O(1) space
    - A = 12 | 20 | 24 | 32 | 48
    - B = 14 | 19 | 20 | 40 | 48



Featured Challenges

See GitHub Archive!

- github.com/csip-uga/archive/tree/master/2021-07-13
- [CSCI 1301+]: Leetcode 1365
- [CSCI 2720+]: Leetcode 88



Conclusion

Questions?

- Find ACM on:
  - CS GroupMe
  - GitHub
  - LinkedIn
  - Instagram
  - Discord
  - Calendar
- Feel free to message me!
  - jeffery.john@uga.edu | ugaacm@uga.edu
  - linkedin.com/in/jefferyjohn