Big-O Notation

A personal appeal

— Growing up I never imagined being a programmer or computer scientist.

— I loved video games and board games maybe too much though..

— Really enjoyed obsessing over how things would play out making decisions.

— Finding the shortest route driving across down, playing with habits and personal routines, puzzles (not actual puzzles, but life puzzles).

Cont...

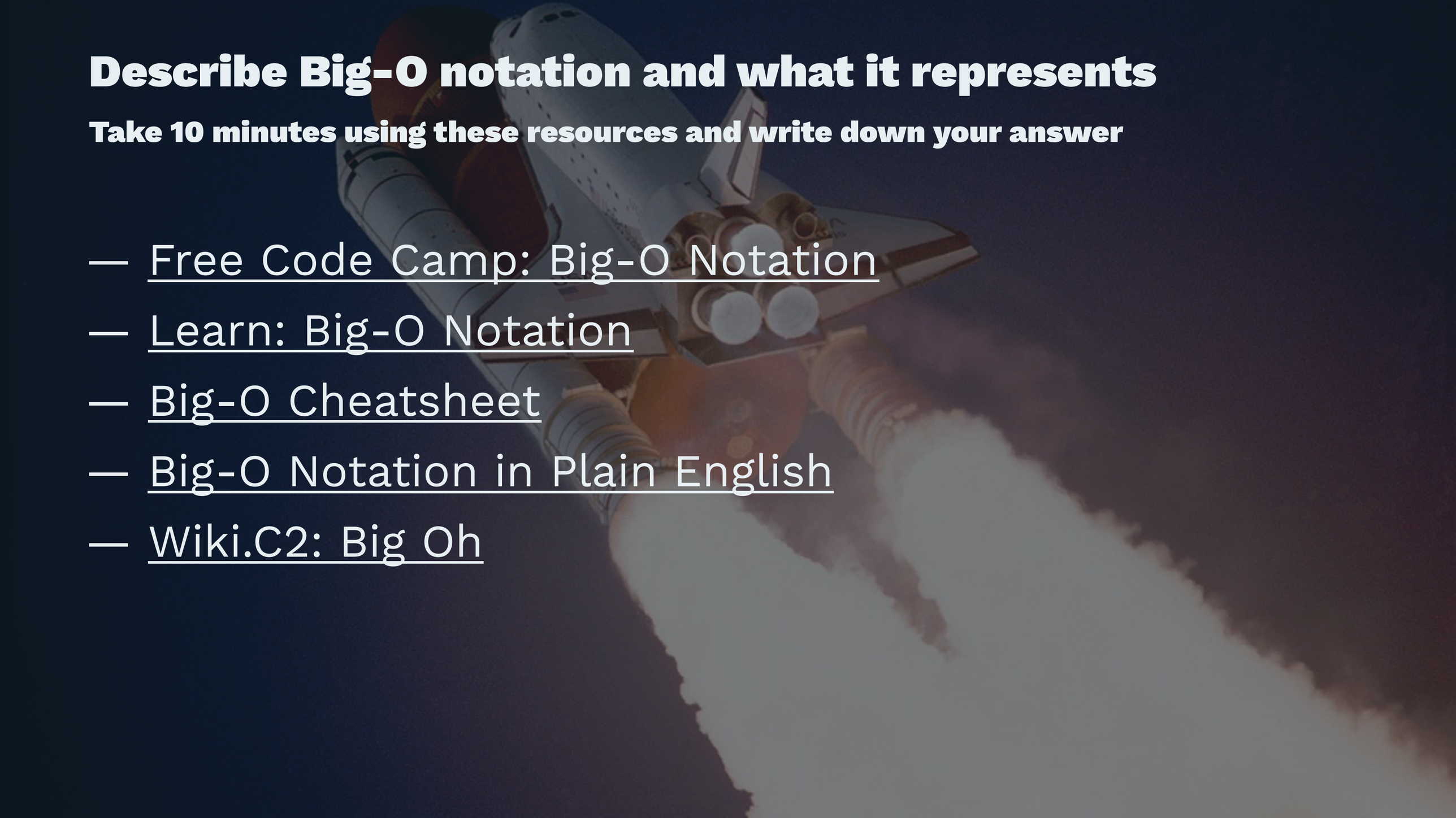
- Maybe you are a resourceful person who finds creative solutions.
- I don’t want to buy a canvas, so I’ll paint on a piece of wood.
- Washing underwear every week is silly, so I’m just not going to wear underwear anymore?
- Bring that mindset to computer science and programming. That is why we are here!

Objectives

- Describe Big-O notation and what it represents
- Compare searching and sorting algorithms with tables and visualizations
- Explain the names and time complexity of the following giving examples of each: O(1), O(log(n)), O(n), O(n log(n)), O(n^2)

Describe Big-O notation and what it represents

Take 10 minutes using these resources and write down your answer



- Free Code Camp: Big-O Notation
- Learn: Big-O Notation
- Big-O Cheatsheet
- Big-O Notation in Plain English
- Wiki.C2: Big Oh

Cold Call

— Q: Describe Big-O notation and what it represents

— A: Big O notation is used to classify algorithms according to how their running time or space requirements grow as the input size grows. "O" is for order or order of the function. This is an upper bound or worst case scenario running time.

Big-O is the worst case

Why is Big-O important?

— Application performance
— Interviews
— Interesting and fun

Compare searching and sorting algorithms with tables and visualizations

We need 8 volunteers in front of the class with their name on whiteboards!

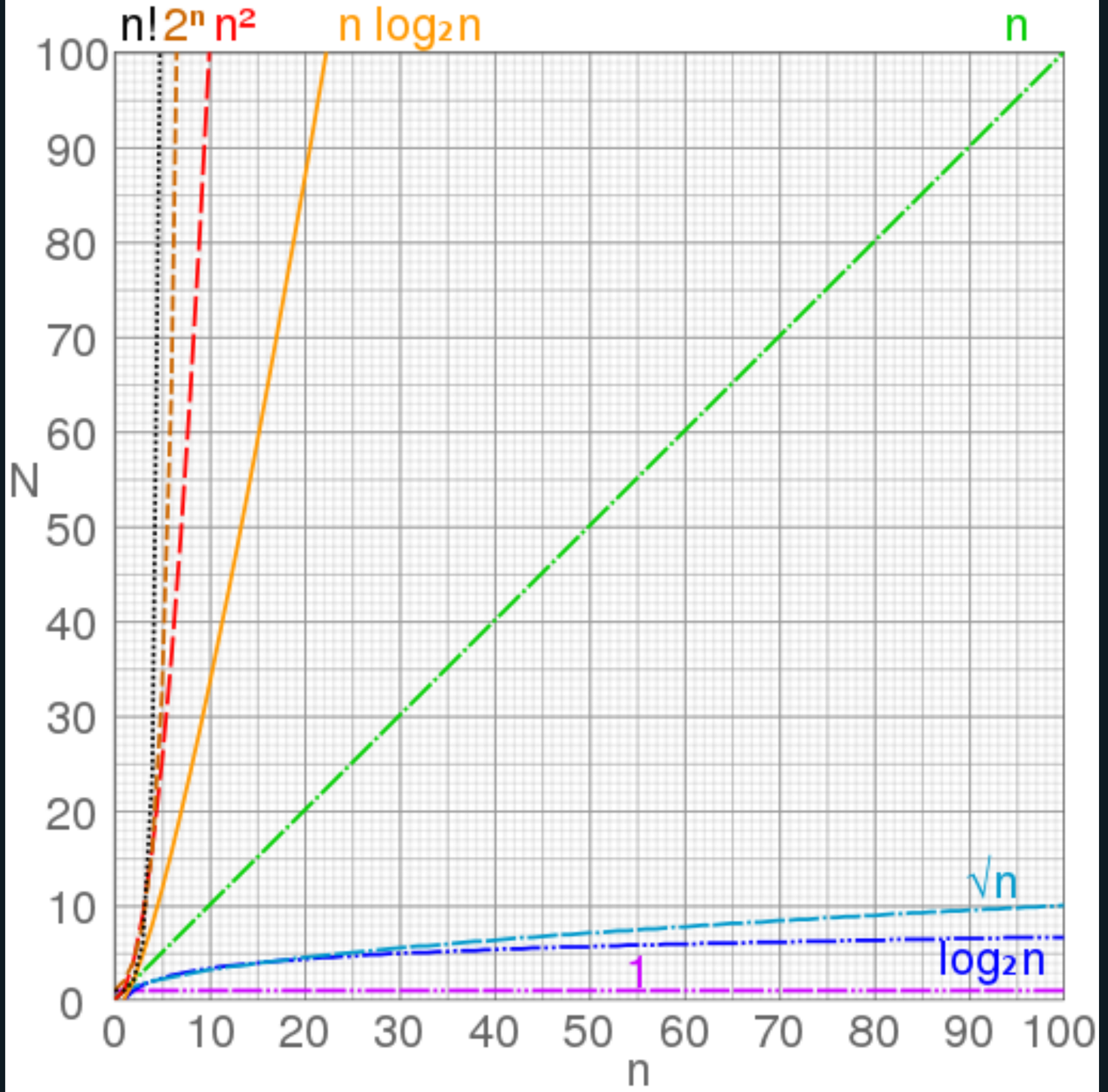
— Unsorted linear searching
— Finding person, not finding person
— How can we make this easier?
— Sorted linear searching
— Can "short-circuit" now, but still worst case is O(n)

Sorting allows us to search more efficiently!

— Binary search
— Finding person, not finding person
— Graph inputs and outputs
— Discuss worst case for binary search being O(log (n))

The simplest sort: bubble sort

— Illustrate swap with students in chair
— Show nested for loop style bubble sort
— Discuss highest "bubbling" to top


n!
2ⁿ
n²
n log₂n
n
√n
log₂n
1
N
n
0
10
20
30
40
50
60
70
80
90
100

Bubble Sort Folk Dance Video


a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
0
1
3
7
2
5
4
8
6
9

CFU

- Q: How many comparisons would take place using bubble sort with an input size of 8?
- Turn and talk with your neighbor
- A: Bubble sort is order n^2, so if n is 8, it would take 64 comparisons.

Explain the names and time complexity of the following:

- O(1)
- O(log(n))
- O(n)
- O(n log(n))
- O(n^2)

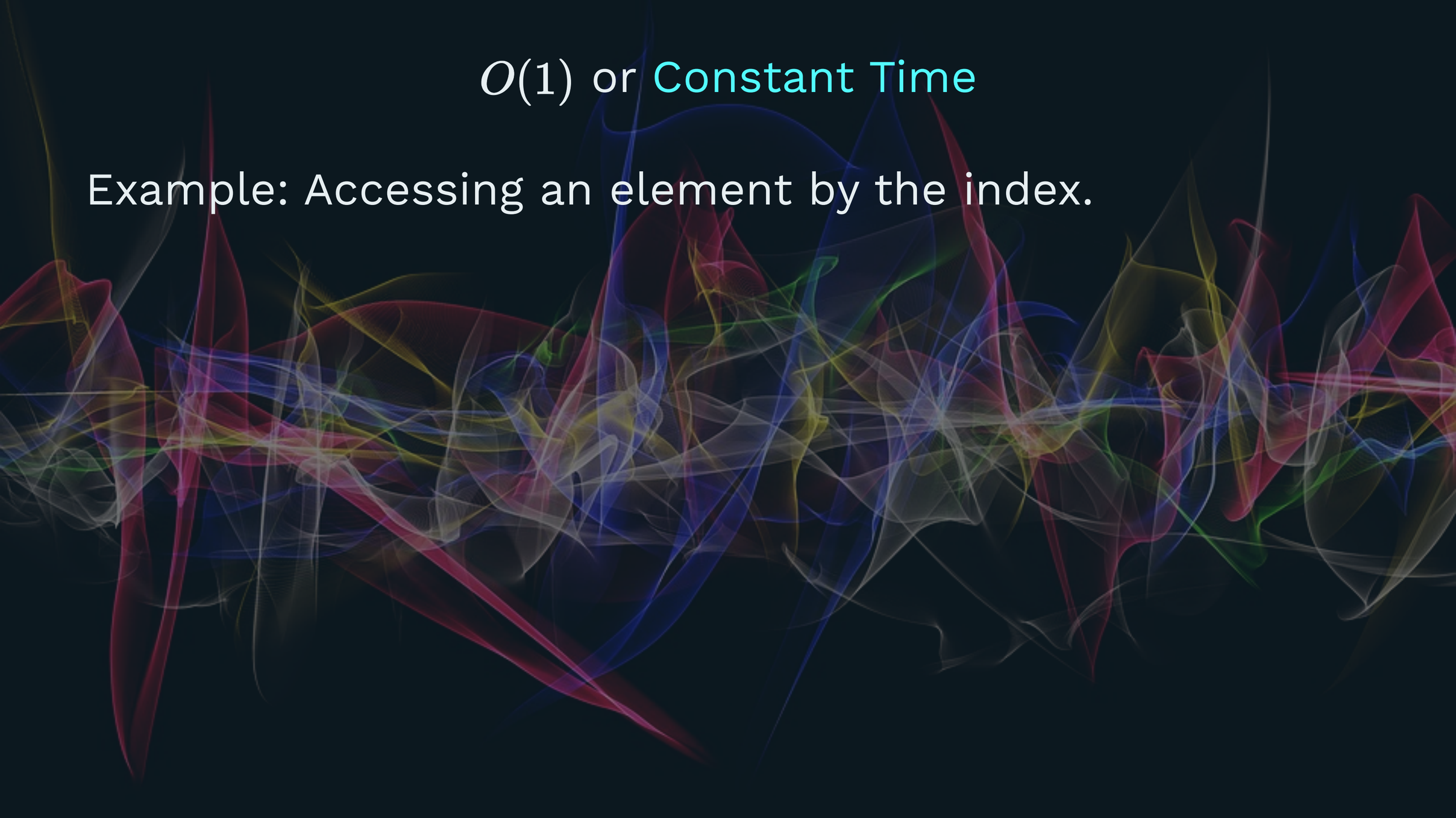
Exercise

Make a copy of this template and fill it out as we go.

Note: Email me (sean.helvey@galvanize.com) your completed notes if you want to attend part 2 of this series, a searching and sorting workshop where we will practice coding these algorithms.

$O(1)$ or Constant Time

Example: Accessing an element by the index.



Still $O(1)$

```
function print50nums() {
  for (var i = 0; i < 50; i++) {
    console.log(i);
  }
}
```

$O(\log n)$ / Logarithmic time

Example: Binary Search

$O(n)$ or Linear Time

Example: Finding an item in an unsorted array.

$O(nlg(n))$ / What about nlg(n)?

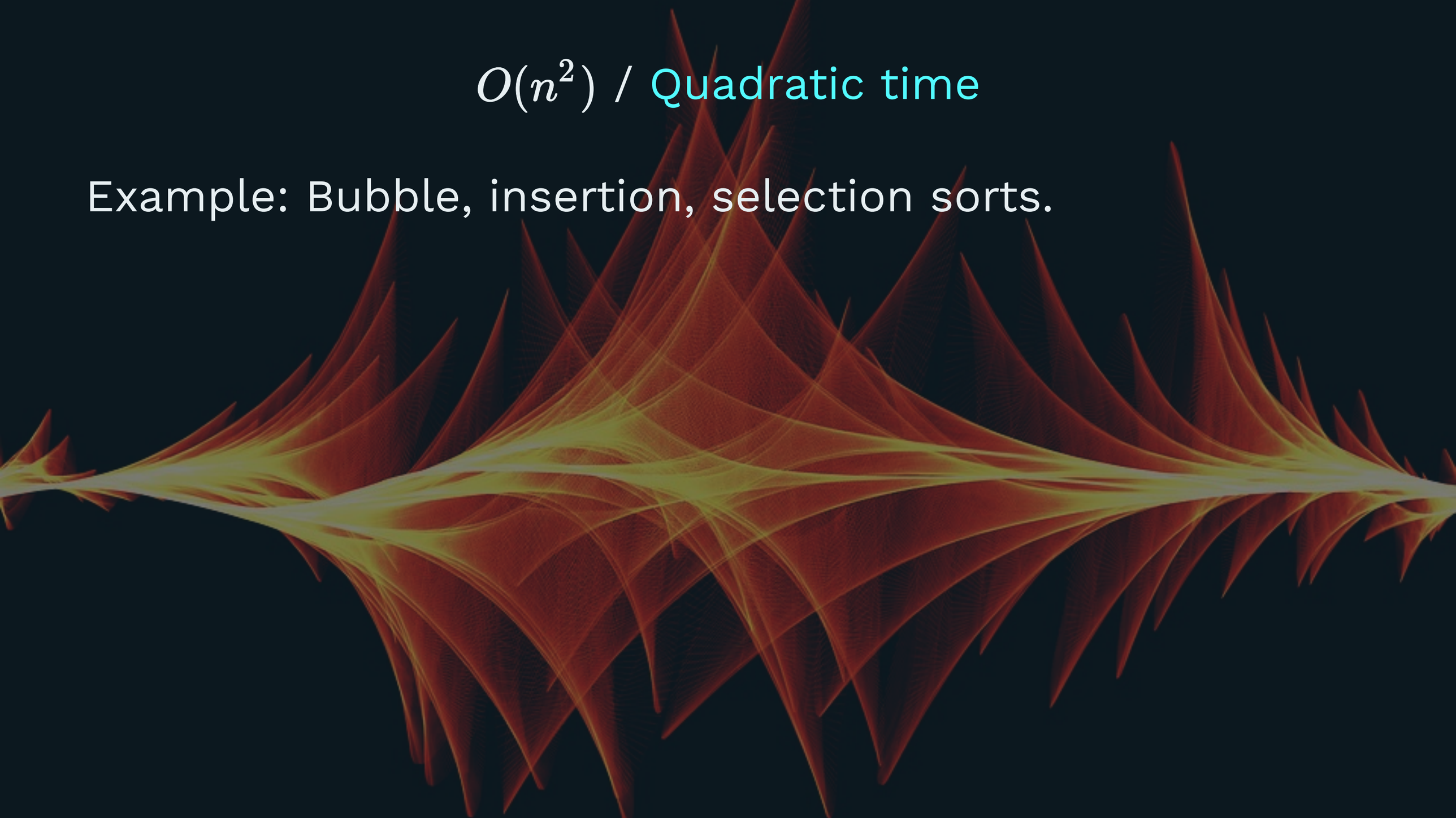
Example: Mergesort & Quicksort.

Mergesort Video


a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
2
4
0
6
8
5
1
7
3
9
b[0]
b[1]
b[2]
b[3]
b[4]
b[5]
b[6]
b[7]
b[8]
b[9]

$O(n^2)$ / Quadratic time

Example: Bubble, insertion, selection sorts.



Objectives

- Describe Big-O notation and what it represents
- Compare searching and sorting algorithms with tables and visualizations
- Explain the names and time complexity of the following giving examples of each: O(1), O(log(n)), O(n), O(n log(n)), O(n^2)