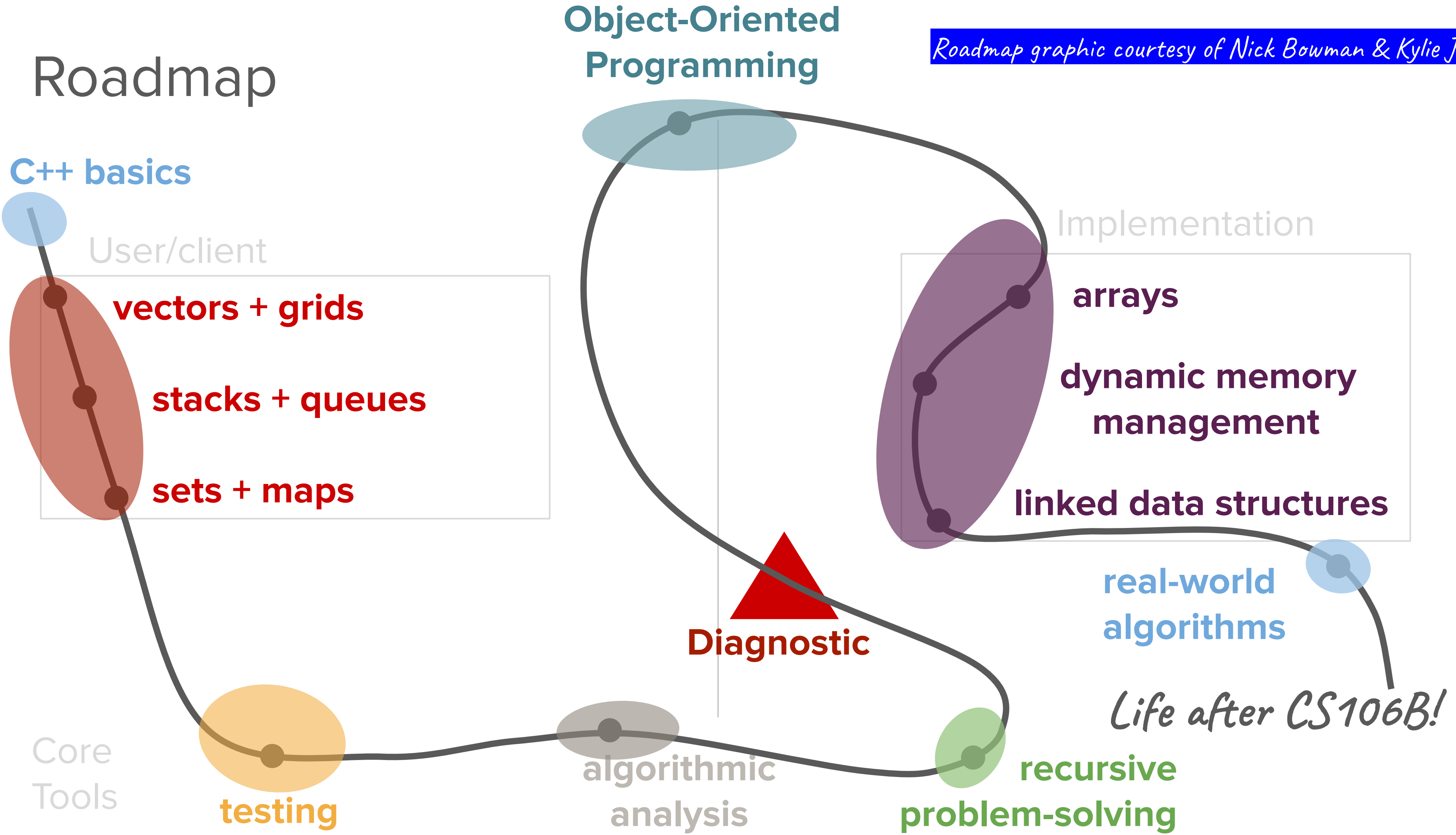
Big-O Notation and Algorithmic Analysis

What do you think makes some algorithms "faster" or "better" than others?

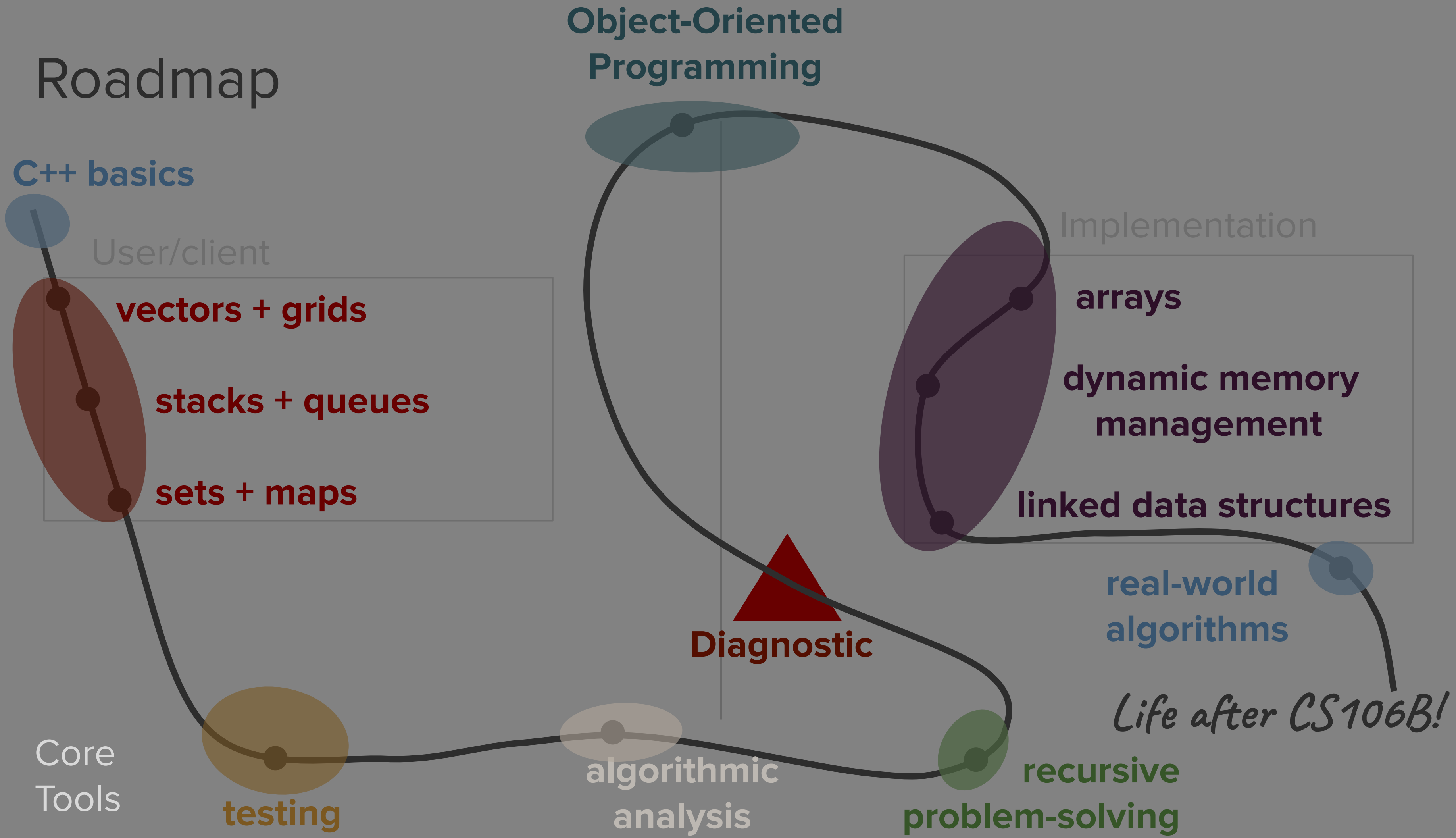
(put your answers the chat)

Roadmap

Roadmap graphic courtesy of Nick Bowman & Kylie Jue



Roadmap


Object-Oriented Programming
C++ basics
User/client
vectors + grids
stacks + queues
sets + maps
Implementation
arrays
dynamic memory management
linked data structures
real-world algorithms
Diagnostic
Life after CS106B!
Core Tools
testing
algorithmic analysis
recursive problem-solving

- There are many ways to solve the same problem.
  How do we **quantitatively** talk about how they compare?
- What might be the **unintentional** impacts of a solution?
- Who will benefit? Will anyone be harmed?
- How will we be able to **test** our solution and measure its efficacy against our goals?
- Who should be invited into the design process?

Today's question

How can we formalize the notion of efficiency for algorithms?

Today's topics

1. Nested Data Structure
2. Big-O Notation
3. Algorithmic Analysis

Pseudocode

Nested Data Structures

Nested Data Structures

- We've already seen one example of nested data structures when we used the **`Queue<Stack<string>>`** to keep track of our search for word ladders.

Nested Data Structures

- We've already seen one example of nested data structures when we used the **`Queue<Stack<string>>`** to keep track of our search for word ladders.

- Nesting data structures (using one ADTs as the data type inside of another ADT) is a great way of organizing data with complex structure.

Nested Data Structures

- We've already seen one example of nested data structures when we used the `Queue<Stack<string>>` to keep track of our search for word ladders.

- Nesting data structures (using one ADTs as the data type inside of another ADT) is a great way of organizing data with complex structure.

- You will thoroughly explore nested data structures (specifically nested Sets and Maps) in Assignment 2!

Nested Data Structures Example

- Imagine we are designing a system to keep track of feeding times for the different animals at a zoo

Nested Data Structures Example

- Imagine we are designing a system to keep track of feeding times for the different animals at a zoo
- Requirements: We need to be able to quickly look up the feeding times associated with an animal if we know it's name. We need to be able to store multiple feeding times for each animal. The feeding times should be stored in the order in which the feedings should happen.

Nested Data Structures Example

- Imagine we are designing a system to keep track of feeding times for the different animals at a zoo
- Requirements: We need to be able to quickly look up the feeding times associated with an animal if we know it's name. We need to be able to store multiple feeding times for each animal. The feeding times should be stored in the order in which the feedings should happen.
- Data Structure Declaration
  - **`Map<string, Vector<string>>`**

Nested Data Structures Example

- Imagine we are designing a system to keep track of feeding times for the different animals at a zoo
- Requirements: We need to be able to quickly look up the feeding times associated with an animal if we know it's name. We need to be able to store multiple feeding times for each animal. The feeding times should be stored in the order in which the feedings should happen.
- Data Structure Declaration
  - `Map<string, Vector<string>>`

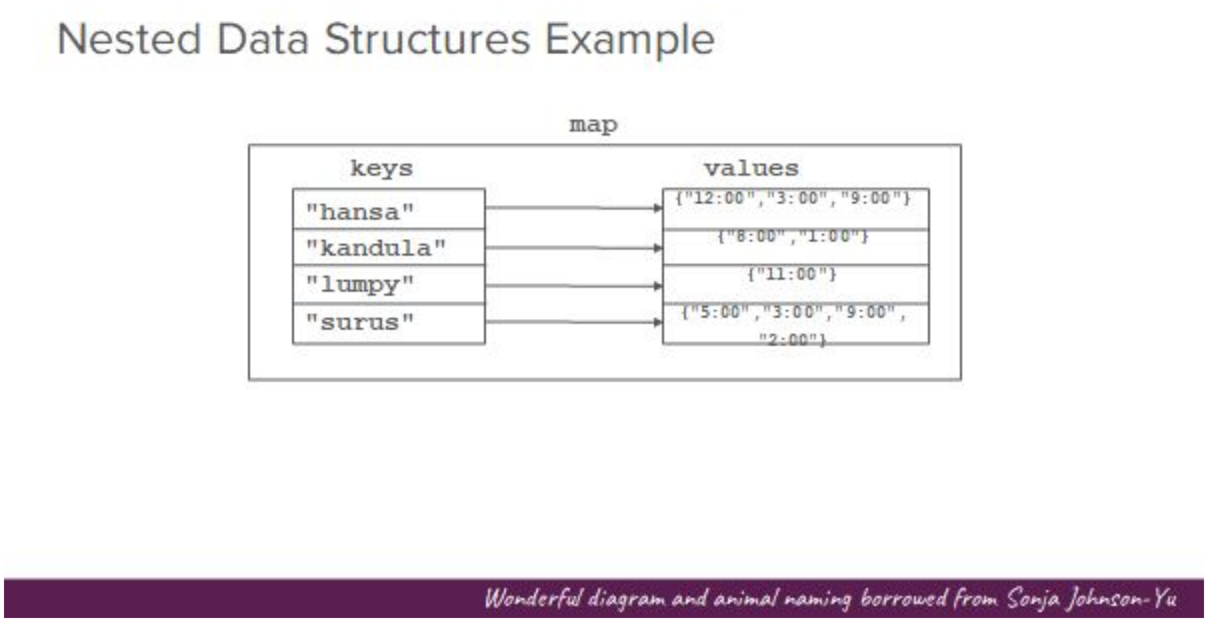
Quick lookup by animal name

Nested Data Structures Example

- Imagine we are designing a system to keep track of feeding times for the different animals at a zoo
- Requirements: We need to be able to quickly look up the feeding times associated with an animal if we know it's name. We need to be able to store multiple feeding times for each animal. The feeding times should be stored in the order in which the feedings should happen.
- Data Structure Declaration
    - **`Map<string, Vector<string>>`**

Store multiple, ordered feeding times per animal

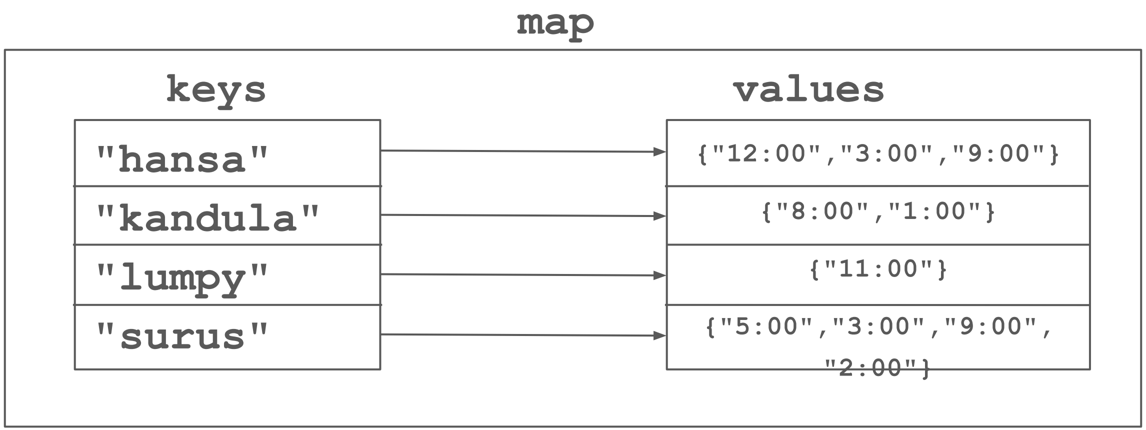
Nested Data Structures Example



Wonderful diagram and animal naming borrowed from Sonja Johnson-Yu



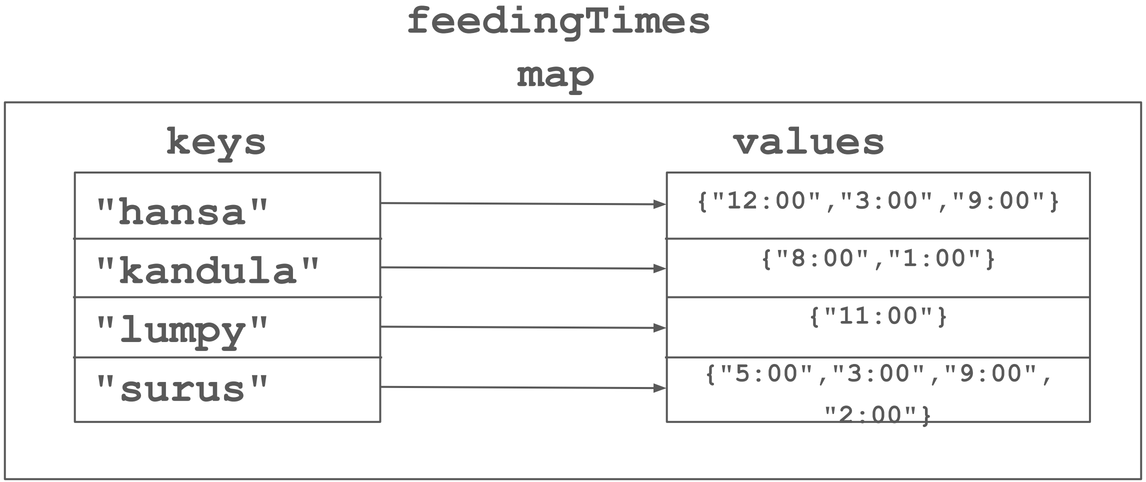
Nested Data Structures Example



How do we use modify the internal values of this map?

Nested Data Structures Example

Goal: We want to add a second feeding time of 4:00 for "lumpy".

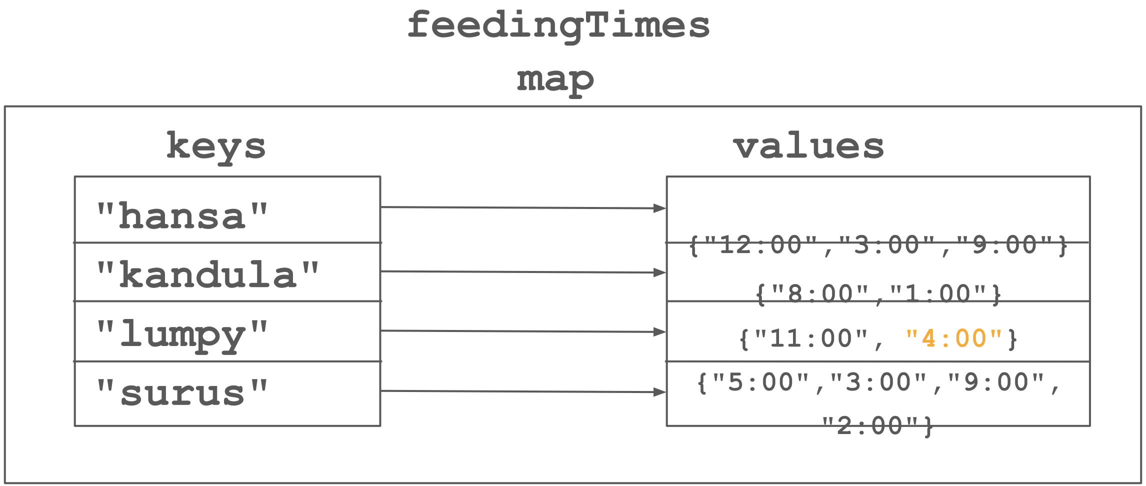


Nested Data Structures Example

Goal: We want to add a second feeding time of 4:00 for "lumpy".

POLL: Which of the following 3 snippets of code will correctly update the state of the map?

1. `feedingTimes["lumpy"].add("4:00");`
2. ```
   Vector<string> times = feedingTimes["lumpy"];
   times.add("4:00");
   ```
3. ```
   Vector<string> times = feedingTimes["lumpy"];
   times.add("4:00");
   feedingTimes["lumpy"] = times;
   ```

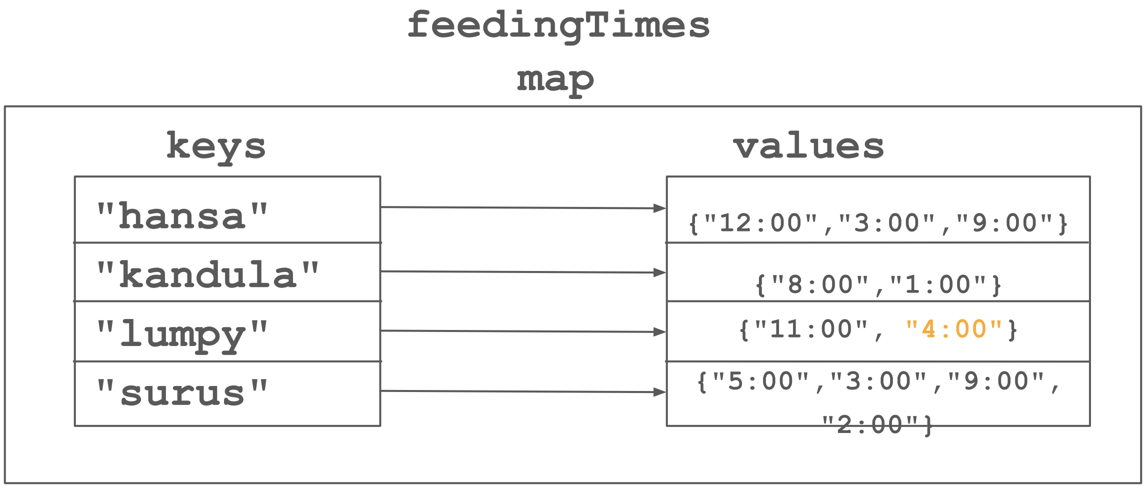


Nested Data Structures Example

Goal: We want to add a second feeding time of 4:00 for "lumpy".

Which of the following three snippets of code will correctly update the state of the map?

1. `feedingTimes["lumpy"].add("4:00");`
2. `Vector<string> times = feedingTimes["lumpy"]; times.add("4:00");`
3. `Vector<string> times = feedingTimes["lumpy"]; times.add("4:00"); feedingTimes["lumpy"] = times;`



`[]` Operator and `=` Operator Nuances

- When you use the [] operator to access an element from a map, you get a reference to the map, which means that any changes you make to the reference will be persistent in the map.
    - `feedingTimes["lumpy"].add("4:00");`

`[]` Operator and `=` Operator Nuances

- When you use the [] operator to access an element from a map, you get a reference to the map, which means that any changes you make to the reference will be persistent in the map.
  - `feedingTimes["lumpy"].add("4:00");`
- However, when you use the = operator to assign the result of the [] operator to a variable, you get a copy of the internal data structure.
  - ```
    Vector<string> times = feedingTimes["lumpy"]; // this makes a copy
    times.add("4:00"); // modifies the copy, not the actual map value!!!
    ```
    ```

`[]` Operator and `=` Operator Nuances

- When you use the [] operator to access an element from a map, you get a reference to the map, which means that any changes you make to the reference will be persistent in the map.
  - `feedingTimes["lumpy"].add("4:00");`
- However, when you use the = operator to assign the result of the [] operator to a variable, you get a copy of the internal data structure.
  - ```
    Vector<string> times = feedingTimes["lumpy"]; // this makes a copy
    times.add("4:00"); // modifies the copy, not the actual map value!!!
    ```
- If you choose to store the internal data structure in a variable, you must do an explicit reassignment to get your changes to persist
  - ```
    Vector<string> times = feedingTimes["lumpy"]; // this makes a copy
    times.add("4:00"); // modifies the copy
    feedingTimes["lumpy"] = times; // stores the modified copy in the map
    ```

Nested ADTs Summary

- Powerful
    - Can express highly structured and complex data
    - Used in many real-world systems

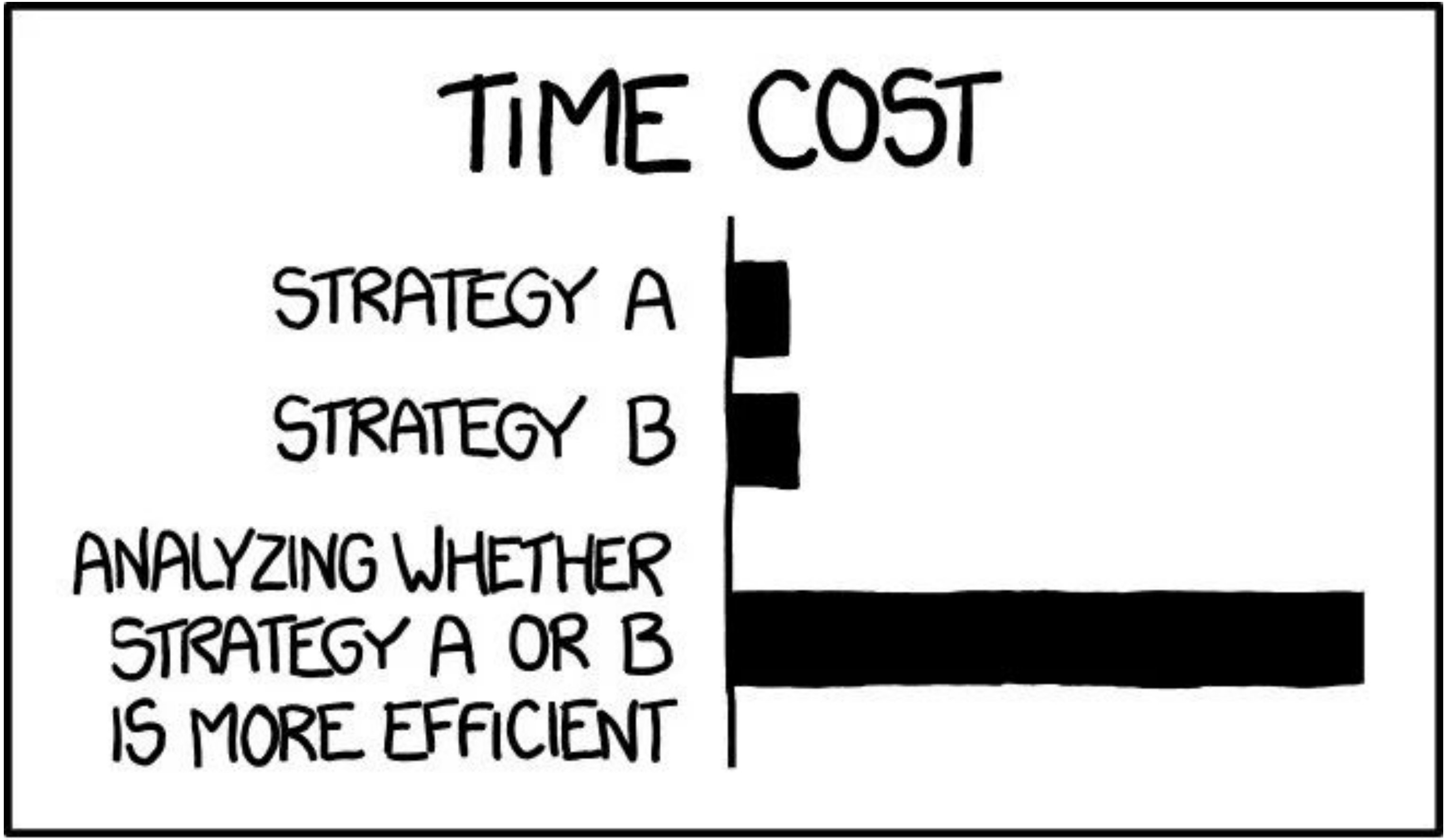
- Tricky
    - With increased complexity comes increased cognitive load in differentiating between the levels of information stored at each level of the nesting
    - Specifically in C++, working with nested data structures can be tricky due to the fact that references and copies show up at different points in time. Follow the correct paradigms presented earlier to stay on track!

Const Reference

- Passing a large object (e.g. a million-element Vector) by value makes a copy, which can take a lot of time.
- Taking parameters by reference avoids making a copy, but risks that the object gets tampered with in the process.
- As a result, it's common to have functions that take objects as parameters take their argument by const reference:
  - The "by reference" part avoids a copy.
  - The "const" (constant) part means that the function can't change that argument.
- For example:

```
void proofreadLongEssay(const string& essay) {
    /* can read, but not change, the essay. */
}
```

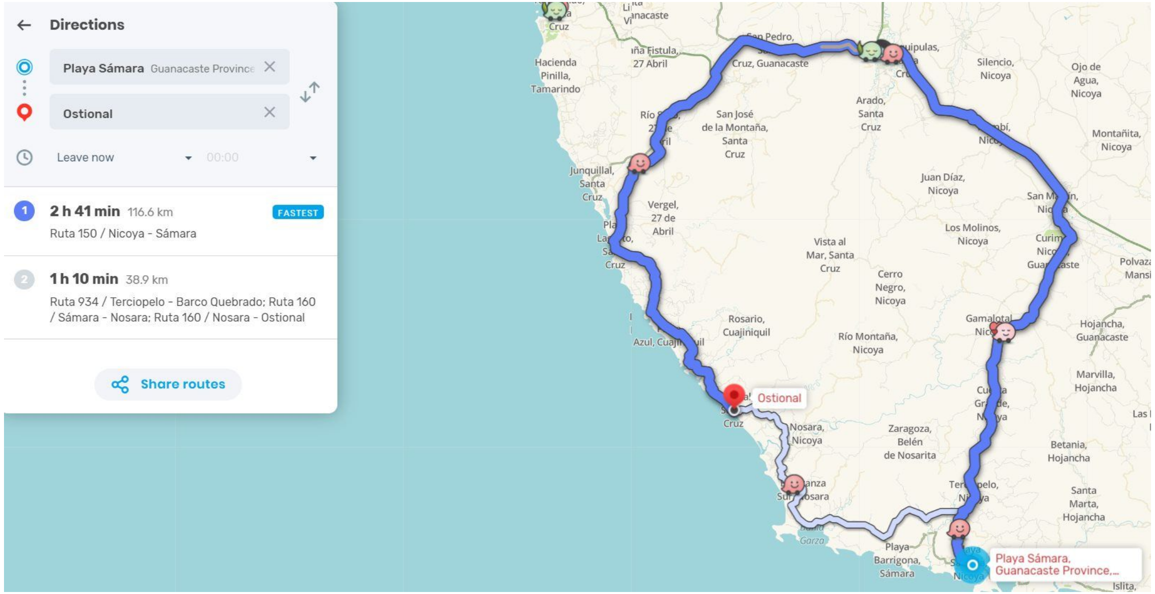
How can we formalize the notion of efficiency for algorithms?



THE REASON I AM SO INEFFICIENT

Why do we care about efficiency?

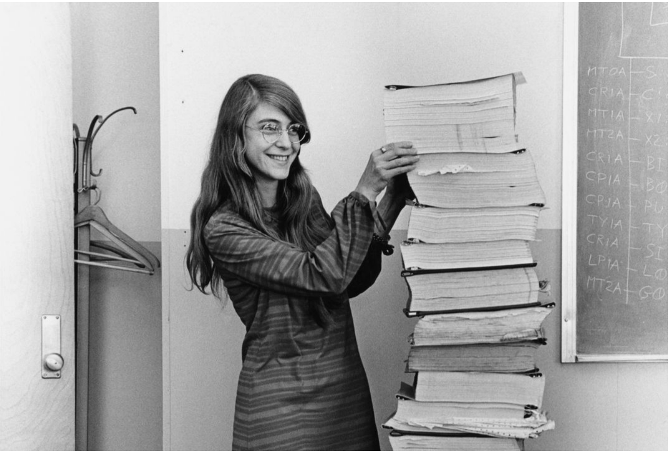
- Implementing inefficient algorithms may make solving certain tasks impossible, even with unlimited resources



Why do we care about efficiency?

- Implementing inefficient algorithms may make solving certain tasks impossible, even with unlimited resources

- Implementing efficient algorithms allows us to solve important problems, often with limited resources available



Source: https://datacenterfrontier.com/year-hyperscale-facebook-growth-innovation/

Why do we care about efficiency?

- Implementing inefficient algorithms may make solving certain tasks impossible, even with unlimited resources

- Implementing efficient algorithms allows us to solve important problems, often with limited resources available

- If we can quantify the efficiency of an algorithm, we can understand and predict its behavior when we apply it to unseen problems

- Efficient algorithms are “green” algorithms – they are better for our climate.

Assignment 1 Redux

- In Assignment 1, you implemented three different algorithms for finding perfect numbers

Assignment 1 Redux

- In Assignment 1, you implemented three different algorithms for finding perfect numbers
  - Exhaustive Search
    - Runtime predictions to find 5th perfect number: Anywhere from 25-100+ days

Assignment 1 Redux

- In Assignment 1, you implemented three different algorithms for finding perfect numbers
  - Exhaustive Search
    - Runtime predictions to find 5th perfect number: Anywhere from 25-100+ days
  - Smarter Search
    - Runtime predictions to find 5th perfect number: Anywhere from a couple minutes to 1 hour

Assignment 1 Redux

- In Assignment 1, you implemented three different algorithms for finding perfect numbers
  - Exhaustive Search
    - Runtime predictions to find 5th perfect number: Anywhere from 25-100+ days
  - Smarter Search
    - Runtime predictions to find 5th perfect number: Anywhere from a couple minutes to 1 hour
  - Euclid's Algorithm
    - Actual runtime to predict 5th perfect number: Less than a second!

Assignment 1 Redux

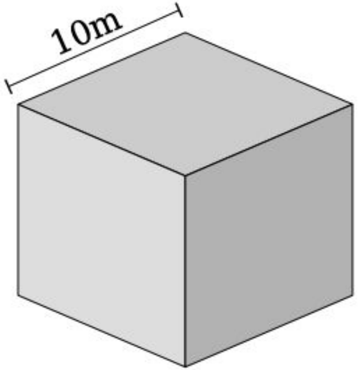
- In Assignment 1, you implemented three different algorithms for finding perfect numbers
  - Exhaustive Search
    - Runtime predictions to find 5th perfect number: Anywhere from 25-100+ days
  - Smarter Search
    - Runtime predictions to find 5th perfect number: Anywhere from a couple minutes to 1 hour
  - Euclid's Algorithm
    - Actual runtime to predict 5th perfect number: Less than a second!
- Core idea: Although each individual experienced dramatically different real runtimes for these three algorithms, there is a clear distinction here between "fast"/"efficient" and "slow"/"inefficient" algorithms

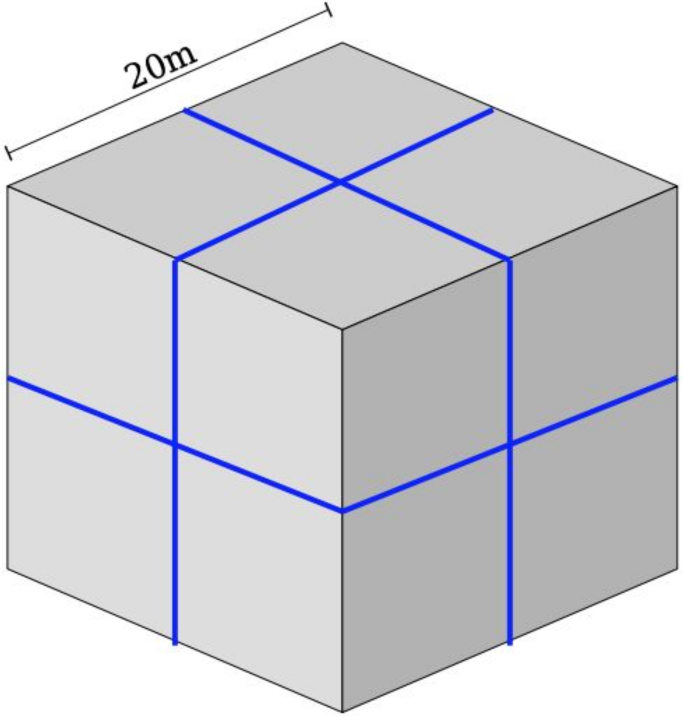
Estimating Quantities

Leveraging Intuition for Estimation

Here are 5 scenarios where you have 2 similar items of different magnitudes, one small and one larger. You know the exact magnitude of the smaller item. Can you predict what the magnitude of the larger item will be based on the intuitive visual relationship?

Example 1

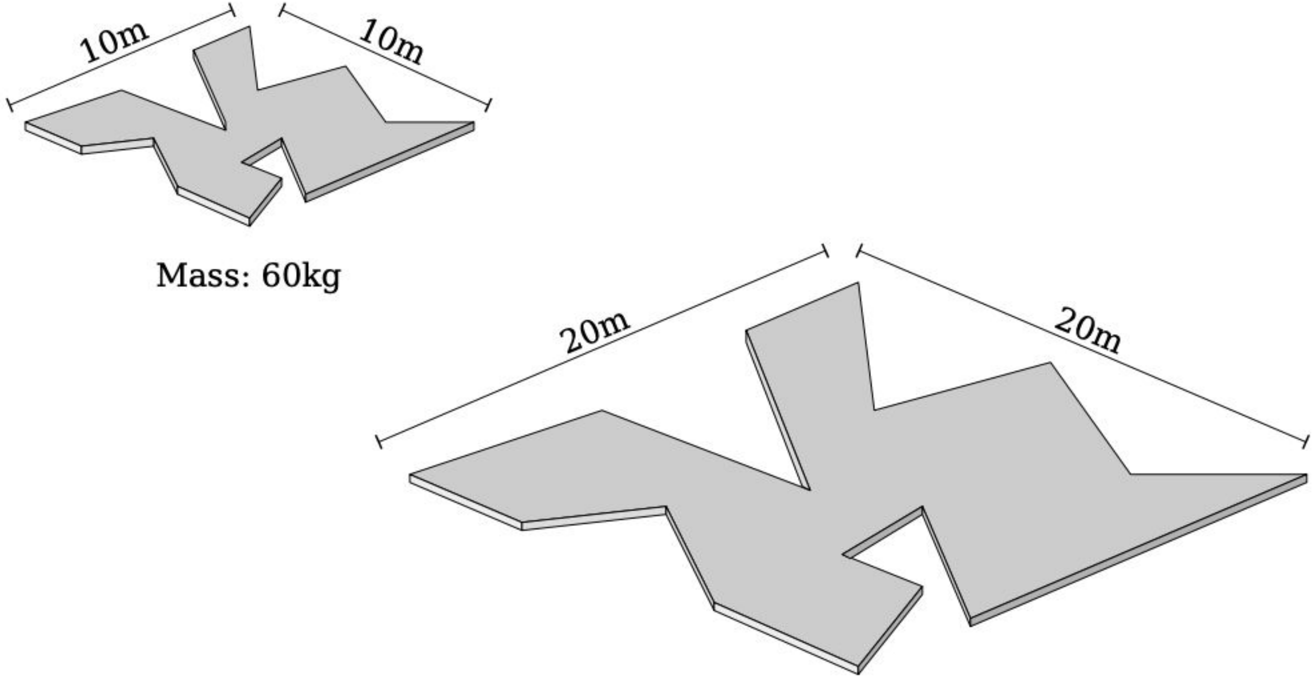




These two cubes are made of the same material.

What’s your best guess for the mass of the second cube?

Example 2

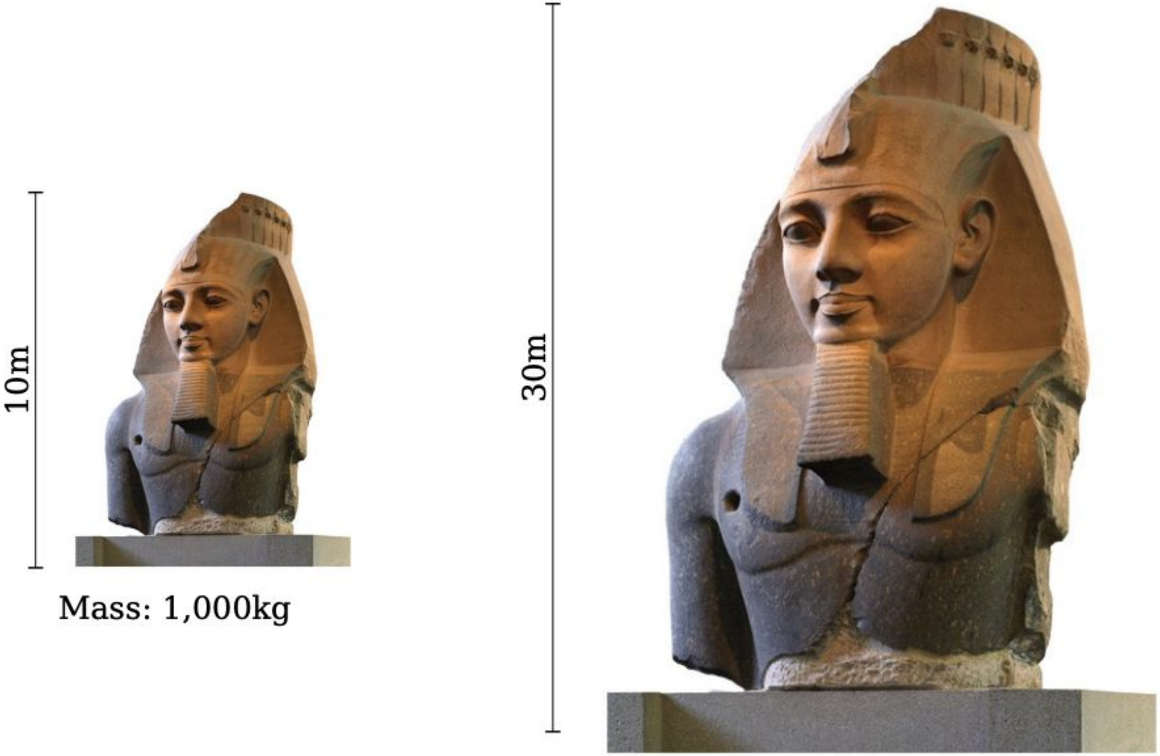


These two square plates are made of the same material.

They have the same thickness.

What's your best guess for the mass of the second square?

Example 3



Mass: 1,000kg

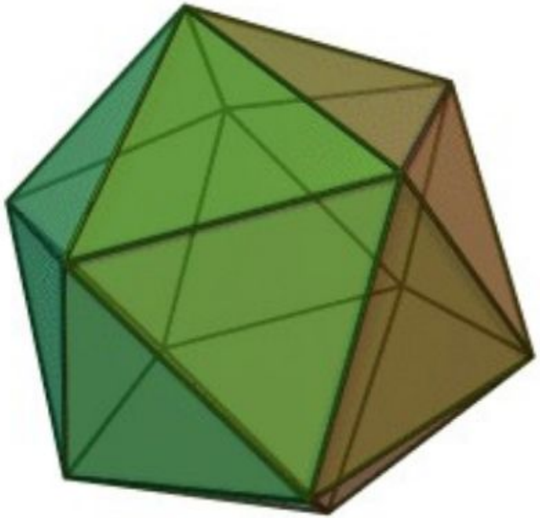
These two statues are made of the same material.

What's your best guess for the mass of the second statue?

Example 4

All sides of each triangle
are 10m long.

Paint required:
90L



All sides of each triangle
are 40m long.

How much paint is needed to paint the surface of the larger icosahedron?

Key Takeaway

Knowing the rate at which some quantity scales allows you to predict its value in the future, even if you don't have an exact formula.

Announcements

Announcements

- Assignment 2 is out! It’s due end of the day on **Wednesday, July 7**.
    - YEAH will be today, 7/1, at 7pm PT. Link is on the course website on the zoom info page.

Big-O Notation

Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.

Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.
- Example:
  - A square of side length $\mathbf{r}$ has area $\mathbf{O(r^2)}$.

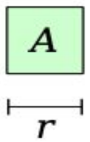
Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.
- Example:
  - A square of side length `r` has area `O(r²)`.

The "O" stands for "on the order of", which is a growth prediction, not an exact formula

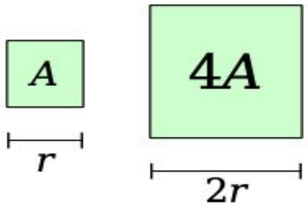
Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.
- Example:
  - A square of side length `r` has area `O(r²)`.



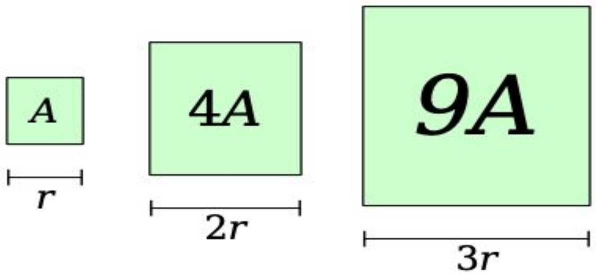
Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.
- Example:
  - A square of side length `r` has area `O(r`^{2}`)`.



Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.
- Example:
    - A square of side length r has area $O(r^2)$.

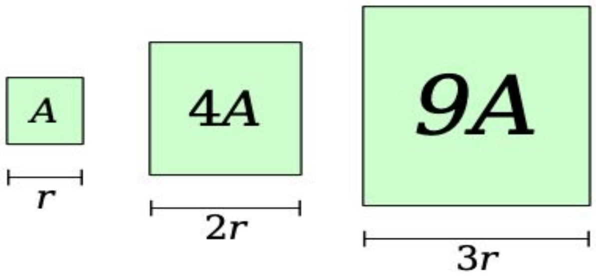


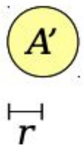
Doubling r increases area 4x

Tripling r increases area 9x

Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.
- Example:
  - A square of side length r has area $O(r^2)$.
  - A circle of radius r has area $O(r^2)$.



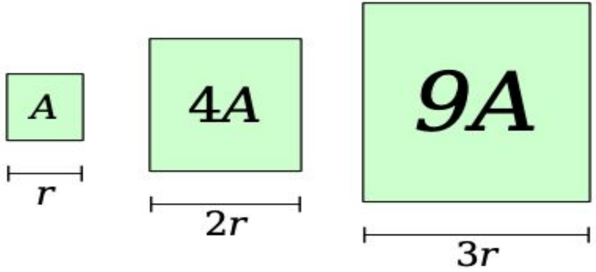


Doubling r increases area 4x

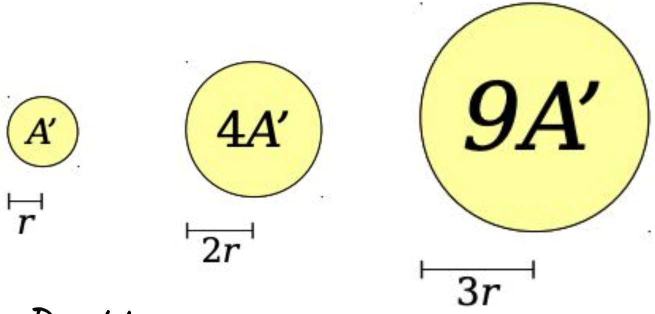
Tripling r increases area 9x

Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.
- Example:
  - A square of side length r has area $O(r^2)$.
  - A circle of radius r has area $O(r^2)$.



Doubling r increases area 4x
Tripling r increases area 9x

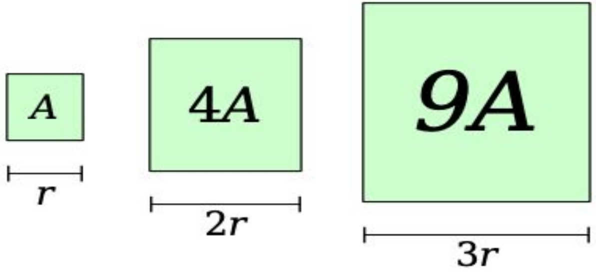


Doubling r increases area 4x
Tripling r increases area 9x

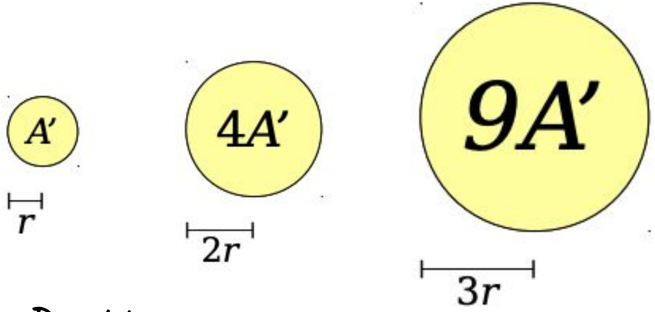
Big-O Notation

- **Big-O notation** is a way of quantifying the rate at which some quantity grows.
- Example:
  - A square of side length r has area $O(r^2)$.
  - A circle of radius r has area $O(r^2)$.

This just says that these quantities grow at the same relative rates. It does not say that they're equal!



Doubling r increases area 4x
Tripling r increases area 9x

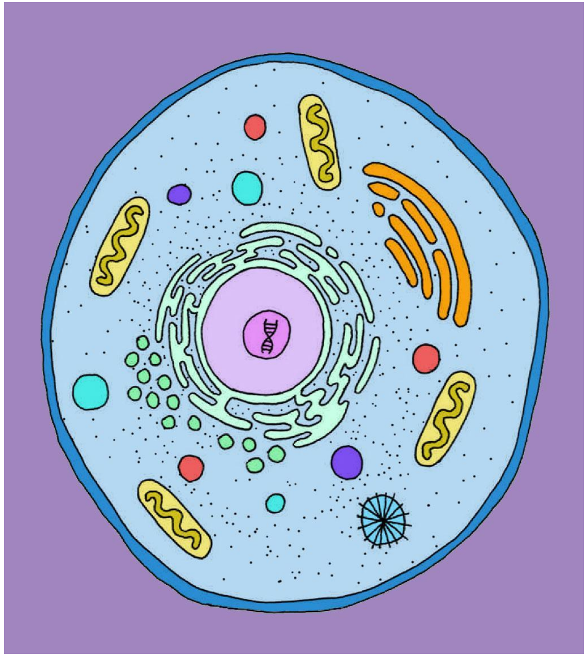


Doubling r increases area 4x
Tripling r increases area 9x

Big-O in the Real World

Big-O Example: Cell Size

- Question: Why are cells tiny?



Big-O Example: Cell Size

- Question: Why are cells tiny?
- Assumption: Cells are spheres

Big-O Example: Cell Size

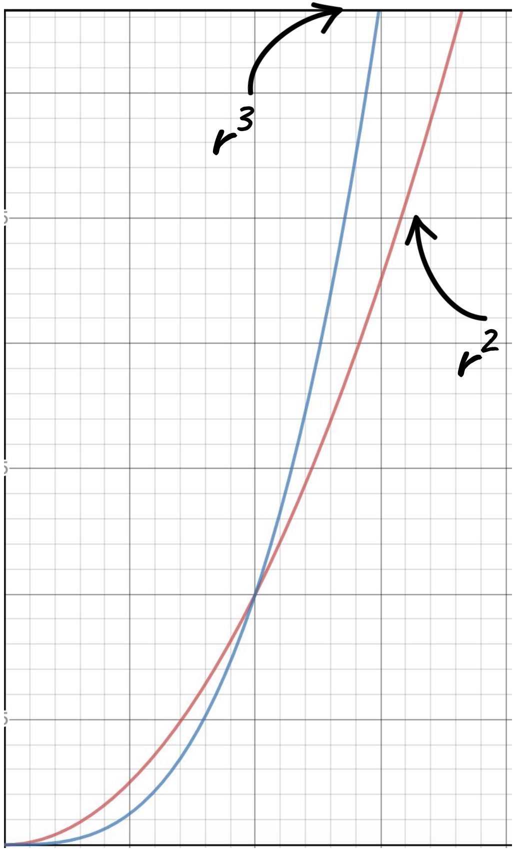
- Question: Why are cells tiny?
- Assumption: Cells are spheres
- A cell absorbs nutrients from its environment through its surface area.
    - Surface area of the cell: $O(r^2)$

Big-O Example: Cell Size

- Question: Why are cells tiny?
- Assumption: Cells are spheres
- A cell absorbs nutrients from its environment through its surface area.
  - Surface area of the cell: $O(r^2)$
- A cell needs to provide nutrients all throughout its volume
  - Volume of the cell: $O(r^3)$

Big-O Example: Cell Size

- Question: Why are cells tiny?
- Assumption: Cells are spheres
- A cell absorbs nutrients from its environment through its surface area.
  - Surface area of the cell: $O(r^2)$
- A cell needs to provide nutrients all throughout its volume
  - Volume of the cell: $O(r^3)$
- As a cell gets bigger, its resource *intake* grows slower than its resource *consumption*, so each part of the cell gets less energy.



Big-O Example: Manufacturing

Big-O Example: Manufacturing

- It costs you some amount of money to produce a cat toy, and there were some one-time expenses to set up the factory.
- What data would you need to gather to estimate the cost of producing ten million cat toys?

Big-O Example: Manufacturing

- You're working at a company producing cat toys. It costs you some amount of money to produce a cat toy, and there were some one-time expenses to set up the factory.
- What data would you need to gather to estimate the cost of producing ten million cat toys?

```
Cost(n) = n × costPerToy + startupCost
```

Big-O Example: Manufacturing

- You’re working at a company producing cat toys. It costs you some amount of money to produce a cat toy, and there were some one-time expenses to set up the factory.
- What data would you need to gather to estimate the cost of producing ten million cat toys?

This term grows as a function of n

```
Cost(n) = n × costPerToy + startupCost
```

Big-O Example: Manufacturing

- You're working at a company producing cat toys. It costs you some amount of money to produce a cat toy, and there were some one-time expenses to set up the factory.
- What data would you need to gather to estimate the cost of producing ten million cat toys?

This term grows as a function of n

This term does not grow

```
Cost(n) = n × costPerToy + startupCost
```

Big-O Example: Manufacturing

- You're working at a company producing cat toys. It costs you some amount of money to produce a cat toy, and there were some one-time expenses to set up the factory.
- What data would you need to gather to estimate the cost of producing ten million cat toys?

This term grows as a function of n

This term does not grow

```
Cost(n) = n × costPerToy + startupCost
        = O(n)
```

Nuances of Big-O

- Big-O notation is designed to capture **the rate at which a quantity grows.** It <u>does not</u> capture information about
    - leading coefficients: the area of a square and a circle are both $O(r^2)$.
    - lower-order terms: there may be other factors contributing to growth that get glossed over.

- However, it's still a **very powerful tool for predicting behavior.**

Analyzing Code

How can we apply Big-O to computer science?

Why runtime isn’t enough

- What is runtime?
    - Runtime is simply the amount of real time it takes for a program to run

Why runtime isn't enough

- What is runtime?
  - Runtime is simply the amount of real time it takes for a program to run

```
[SimpleTest] ---- Tests from main.cpp -----
[SimpleTest] starting (PROVIDED_TEST, line  36) timing vectorMax on 10,00...  =  Correct
    Line 42 Time vectorMax(v) (size =10000000) completed in    0.268 secs
    Line 43 Time vectorMax(v) (size =10000000) completed in    0.264 secs
    Line 44 Time vectorMax(v) (size =10000000) completed in    0.269 secs
You passed 1 of 1 tests. Keep it up!
```

Nick's 2012 MacBook

Why runtime isn’t enough

- What is runtime?
    - Runtime is simply the amount of real time it takes for a program to run

```
[SimpleTest] ---- Tests from main.cpp -----
[SimpleTest] starting (PROVIDED_TEST, line  36) timing vectorMax on 10,00...  =  Correct
    Line 42 Time vectorMax(v) (size =10000000) completed in    0.268 secs
    Line 43 Time vectorMax(v) (size =10000000) completed in    0.264 secs
    Line 44 Time vectorMax(v) (size =10000000) completed in    0.269 secs
You passed 1 of 1 tests. Keep it up!
```

Nick's 2012 MacBook

```
[SimpleTest] ---- Tests from main.cpp -----
[SimpleTest] starting (PROVIDED_TEST, line  36) timing vectorMax on 20,00...  =  Correct
    Line 42 Time vectorMax(v) (size =10000000) completed in    0.181 secs
    Line 43 Time vectorMax(v) (size =10000000) completed in    0.181 secs
    Line 44 Time vectorMax(v) (size =10000000) completed in    0.183 secs
You passed 1 of 1 tests. Que bien!
```

Ed's powerful computers

Why runtime isn't enough

- Measuring wall-clock runtime is less than ideal, since
  - It depends on what computer you're using,
  - What else is running on that computer,
  - Whether that computer is conserving power,
  - Etc.

Why runtime isn’t enough

- Measuring wall-clock runtime is less than ideal, since
  - It depends on what computer you’re using,
  - What else is running on that computer,
  - Whether that computer is conserving power,
  - Etc.

- Worse, **individual runtimes can’t predict future runtimes**.

Why runtime isn’t enough

- Measuring wall-clock runtime is less than ideal, since
  - It depends on what computer you’re using,
  - What else is running on that computer,
  - Whether that computer is conserving power,
  - Etc.

- Worse, **individual runtimes can’t predict future runtimes**.

- Let's develop a computer-independent efficiency metric using big-O!

Analyzing Code: `vectorMax()`

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Assume any individual statement takes one unit of time to execute.

If the input **`Vector`** *has* **`n`** *elements, how many time units will this code take to run?*

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total time based on # of repetitions

1 time unit

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total time based on # of repetitions

1 time unit

1 time unit

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total time based on # of repetitions

1 time unit

1 time unit

1 time unit

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total time based on # of repetitions

1 time unit

1 time unit

1 time unit

N time units

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total time based on # of repetitions

1 time unit

1 time unit

1 time unit

N+1 time units

N-1 time units

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total time based on # of repetitions

1 time unit
1 time unit
1 time unit
N time units
N-1 time units
N-1 time units

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total time based on # of repetitions

1 time unit
1 time unit
1 time unit
N time units
N-1 time units
N-1 time units
(up to) N-1 time units

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total time based on # of repetitions

1 time unit
1 time unit
1 time unit
N time units
N-1 time units
N-1 time units
(up to) N-1 time units
1 time unit

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total amount of time

$4N + 1$

vectorMax()

```
int vectorMax(Vector<int> &v) {
  int currentMax = v[0];
  int n = v.size();
  for (int i = 1; i < n; i++) {
      if (currentMax < v[i]) {
          currentMax = v[i];
      }
  }
  return currentMax;
}
```

Total amount of time

O(n)

More practical: Doubling the size of the input roughly doubles the runtime. Therefore, the input and runtime have a linear (O(n)) relationship.

Analyzing Code:
`printStars()`

printStars()

```
void printStars(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << '*' << endl;
        }
    }
}
```

If n = 5

printStars()

```
void printStars(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << '*' << endl;
        }
    }
}
```

printStars()

```
void printStars(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << '*' << endl;
        }
    }
}
```

printStars()

```
void printStars(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            // do a fixed amount of work
        }
    }
}
```

printStars()

```
void printStars(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            // do a fixed amount of work
        }
    }
}
```

printStars()

```
void printStars(int n) {
    for (int i = 0; i < n; i++) {

        // do O(n) time units of work

    }
}
```

printStars()

```
void printStars(int n) {
    for (int i = 0; i < n; i++) {

        // do O(n) time units of work

    }
}
```

printStars()

```
void printStars(int n) {

    // do O(n²) time units of work

}
```

printStars()

```
void printStars(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << '*' << endl;
        }
    }
}
```

$O(n^2)$

A final analyzing code example

hmmThatsStrange()

```
void hmmThatsStrange(int n) {
    cout << "Mirth and Whimsy" << n << endl;
}
```

The runtime is completely independent of the value **n**.

hmmThatsStrange()

```
void hmmThatsStrange(int n) {
    cout << "Mirth and Whimsy" << n << endl;
}
```

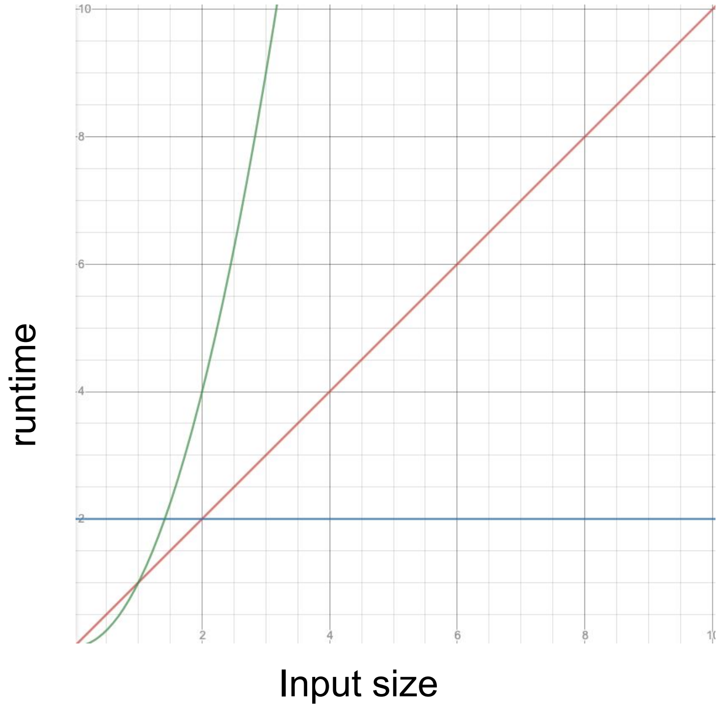
hmmThatsStrange()

```
void hmmThatsStrange(int n) {
    cout << "Mirth and Whimsy" << n << endl;
}
```

O(1)

Efficiency Categorizations So Far

- Constant Time – O(1)
  - Super fast, this is the best we can hope for!
  - example: Euclid's Algorithm for Perfect Numbers
- Linear Time – O(n)
  - This is okay, we can live with this
- Quadratic Time – O(n^2)
  - This can start to slow down really quickly
  - example: Exhaustive Search for Perfect Numbers



Applying Big-O to ADTs

ADT Big-O Matrix

- Vectors
  - `.size() - O(1)`
  - `.add() - O(1)`
  - `v[i] - O(1)`
  - `.insert() - O(n)`
  - `.remove() - O(n)`
  - `.clear() - O(n)`
  - `traversal - O(n)`
- Grids
  - `.numRows()/.numCols() - O(1)`
  - `g[i][j] - O(1)`
  - `.inBounds() - O(1)`
  - **traversal - O(n^2)**

ADT Big-O Matrix

- Vectors
  - `.size() – O(1)`
  - `.add() – O(1)`
  - `v[i] – O(1)`
  - `.insert() – O(n)`
  - `.remove() – O(n)`
  - `.clear() - O(n)`
  - `traversal – O(n)`
- Grids
  - `.numRows()/.numCols() – O(1)`
  - `g[i][j] – O(1)`
  - `.inBounds() – O(1)`
  - `traversal` – O(n^2)
- Queues
  - `.size() – O(1)`
  - `.peek() – O(1)`
  - `.enqueue() – O(1)`
  - `.dequeue() – O(1)`
  - `.isEmpty() – O(1)`
  - `traversal – O(n)`
- Stacks
  - `.size() – O(1)`
  - `.peek() – O(1)`
  - `.push() – O(1)`
  - `.pop() – O(1)`
  - `.isEmpty() – O(1)`
  - `traversal – O(n)`

ADT Big-O Matrix

- Vectors
  - `.size() - O(1)`
  - `.add() - O(1)`
  - `v[i] - O(1)`
  - `.insert() - O(n)`
  - `.remove() - O(n)`
  - `.clear() - O(n)`
  - `traversal - O(n)`
- Grids
  - `.numRows()/.numCols() - O(1)`
  - `g[i][j] - O(1)`
  - `.inBounds() - O(1)`
  - `traversal` - $O(n^2)$
- Queues
  - `.size() - O(1)`
  - `.peek() - O(1)`
  - `.enqueue() - O(1)`
  - `.dequeue() - O(1)`
  - `.isEmpty() - O(1)`
  - `traversal - O(n)`
- Stacks
  - `.size() - O(1)`
  - `.peek() - O(1)`
  - `.push() - O(1)`
  - `.pop() - O(1)`
  - `.isEmpty() - O(1)`
  - `traversal - O(n)`
- Sets
  - `.size() - O(1)`
  - `.isEmpty() - O(1)`
  - `.add() - ???`
  - `.remove() - ???`
  - `.contains() - ???`
  - `traversal - O(n)`
- Maps
  - `.size() - O(1)`
  - `.isEmpty() - O(1)`
  - `m[key] - ???`
  - `.contains() - ???`
  - `traversal - O(n)`

ADT Big-O Matrix

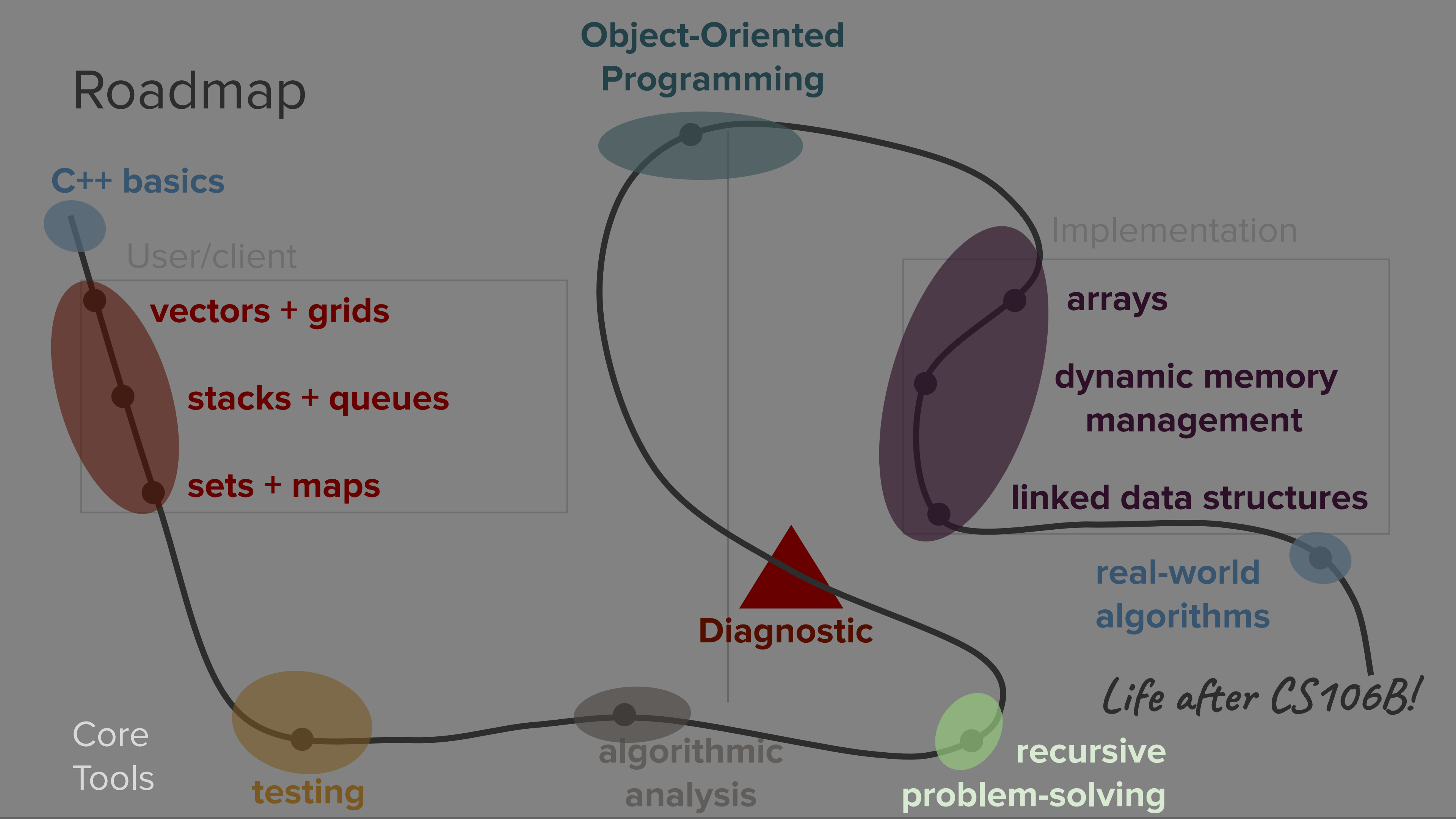
- Vectors
  - `.size() – O(1)`
  - `.add() – O(1)`
  - `v[i] – O(1)`
  - `.insert() – O(n)`
  - `.remove() – O(n)`
  - `.clear() – O(n)`
  - `traversal – O(n)`
- Grids
  - `.numRows()/.numC`
    `– O(1)`
  - `g[i][j] – O(1)`
  - `.inBounds() – O(1)`
  - traversal – $O(n^2)$

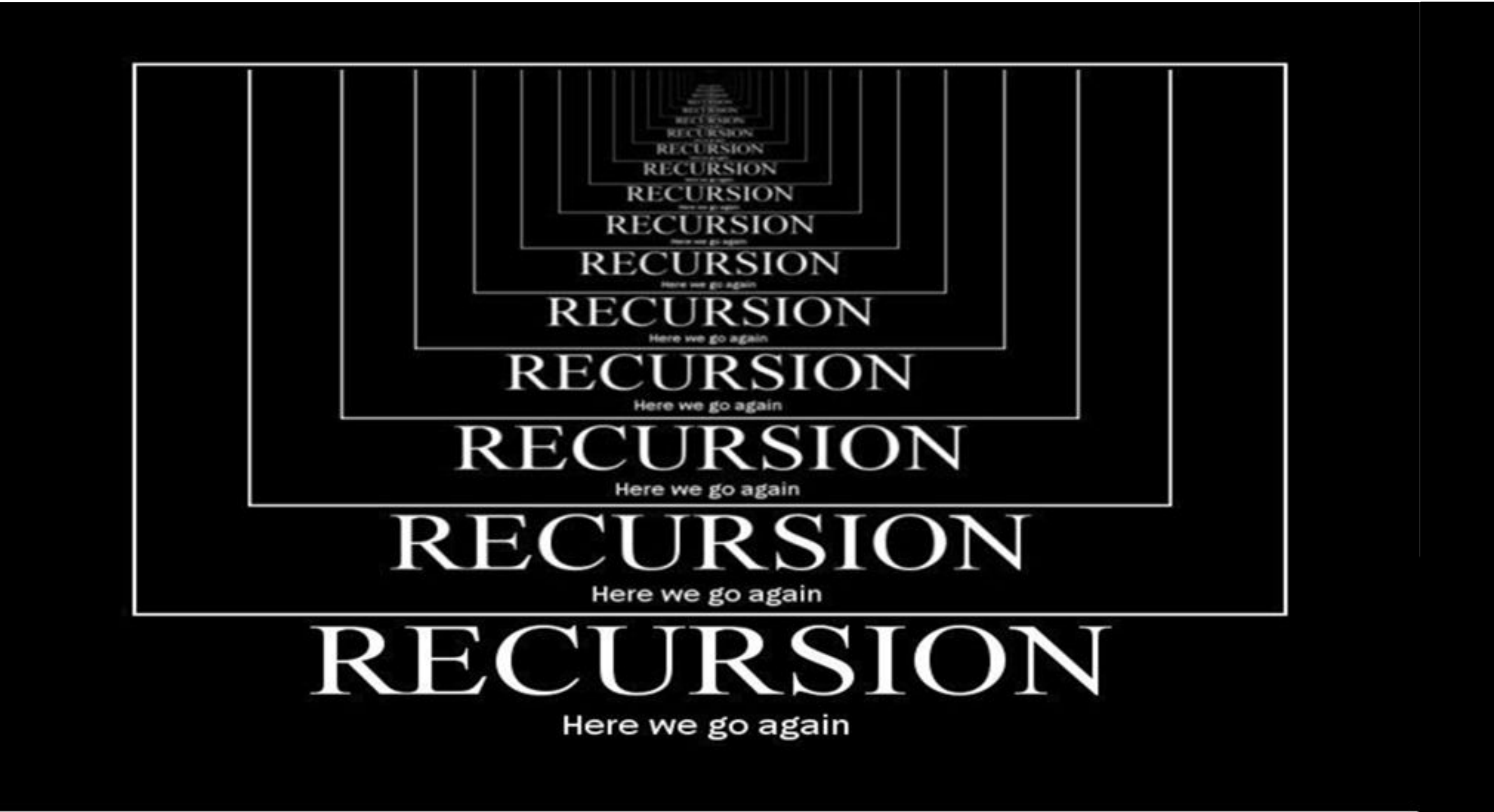
- Queues
  - `size() – O(1)`
  - `.pop() – O(1)`
  - `.isEmpty() – O(1)`
  - `traversal – O(n)`

- Sets
  - `size() – O(1)`
  - `Empty() – O(1)`
  - `d() – ???`
  - `move() – ???`
  - `ntains() – ???`
  - `versal – O(n)`
- aps
  - `ze() – O(1)`
  - `Empty() – O(1)`
  - `key] – ???`
  - `.contains() – ???`
  - `traversal – O(n)`

How can we achieve faster than O(n) runtime when searching/storing n elements?

What’s next?


Roadmap
C++ basics
User/client
vectors + grids
stacks + queues
sets + maps
Object-Oriented Programming
Implementation
arrays
dynamic memory management
linked data structures
Diagnostic
real-world algorithms
Life after CS106B!
Core Tools
testing
algorithmic analysis
recursive problem-solving


RECURSION
RECURSION
RECURSION
RECURSION
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again