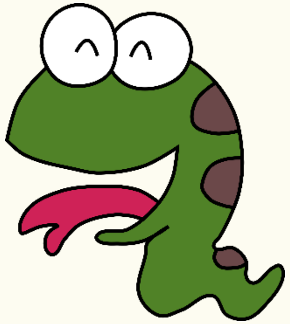


Python Programming

We will be starting shortly

In the meantime, sit back and relax!

(Slides for today’s lecture are on Scientia)

Python Programming

Warning!

We will start recording this session now!

Also, any messages in the text chat will remain on MS Teams even after the session

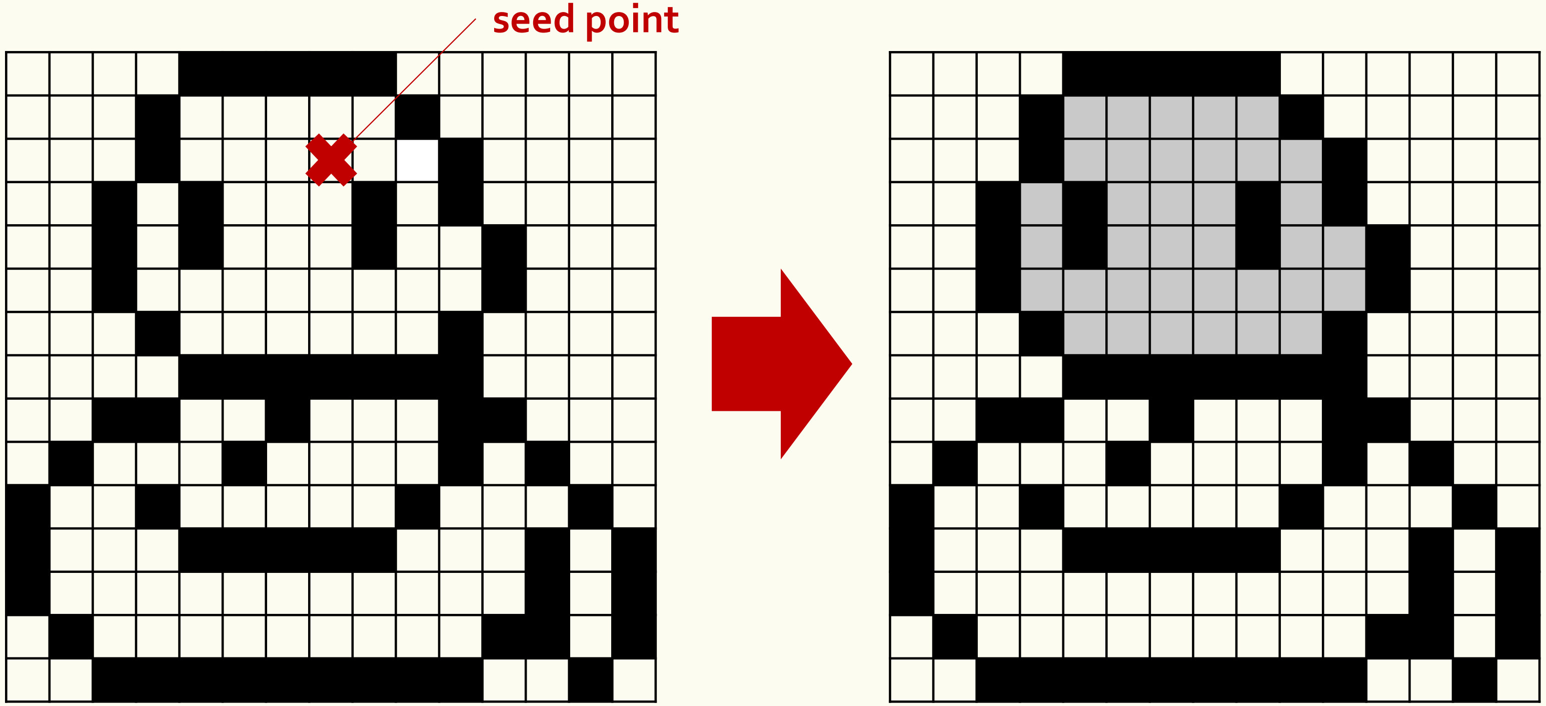
Quiz

What is a snake’s favourite chart?

Coursework released!

Coursework 1

- 4% of final module grade
- Deadline Friday 15th 7pm BST



Submission

- Tasks
  - Complete the fill() function
  - Write some test cases
  - Describe your algorithm (keep it brief!)

- Iterative or recursive solutions both accepted
  - But caution if you are using recursion (Python has a default recursion limit of 1000)

Gitlab Repo

- Most of you should have already received your repo.

- Specifications on CATe and Scientia.

- Submit ‘readme.pdf’ on CATe
- Submit your commit hash via LabTS

- LabTS tests are just to make sure that your code works on the server!
  - You should still test your own code!

Previously...

```
<filter_with_list>,    list size 10,        0.0008445 seconds
<filter_with_dict>,    list size 10,        0.0008762 seconds
<filter_with_set>,     list size 10,        0.0003818 seconds
<filter_with_list>,    list size 100,       0.031647 seconds
<filter_with_dict>,    list size 100,       0.0032972 seconds
<filter_with_set>,     list size 100,       0.0012386 seconds
<filter_with_list>,    list size 600,       0.96294 seconds
<filter_with_dict>,    list size 600,       0.020129 seconds
<filter_with_set>,     list size 600,       0.0093079 seconds
<filter_with_list>,    list size 4500,      10.718 seconds
<filter_with_dict>,    list size 4500,      0.1193 seconds
<filter_with_set>,     list size 4500,      0.03865 seconds
<filter_with_list>,    list size 30000,     85.121 seconds
<filter_with_dict>,    list size 30000,     0.83646 seconds
<filter_with_set>,     list size 30000,     0.22502 seconds
```

Writing efficient code

that scales well with input size

Searching algorithms

because AI involves lots of searching!

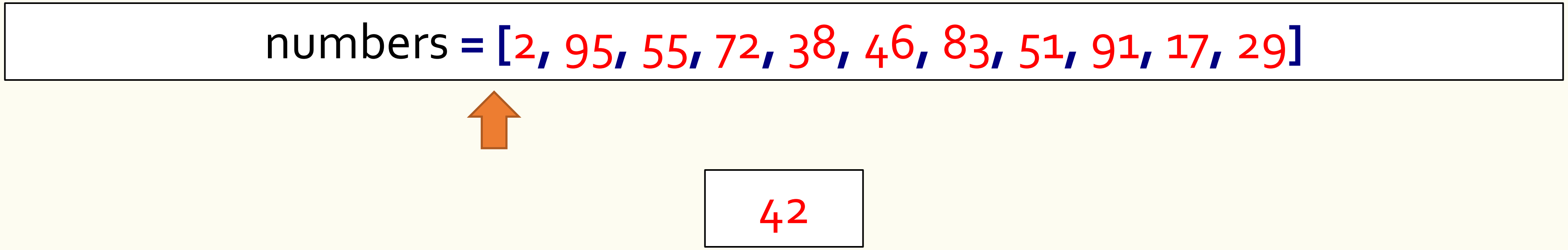
```
numbers = [2, 95, 55, 72, 38, 46, 83, 51, 91, 17, 29]
```

42

```
42 in numbers
```

```
numbers.index(42)
```

Sequential/linear search



numbers = [2, 95, 55, 72, 38, 46, 83, 51, 91, 17, 29]

42

```
def sequential_search(query, items):
    found = False
    for item in items:
        if item == query:
            found = True

    return found

sequential_search(42, numbers)
```

Time Complexity

Worst case: $O(n)$

Best case: $\Omega(n)$

Sequential/linear search

numbers = [2, 95, 55, 72, 38, 46, 83, 51, 91, 17, 29]

42

```python
def sequential_search(query, items):
    found = False
    for item in items:
        if item == query:
            found = True
            break
    return found

sequential_search(42, numbers)
```

Time Complexity

Worst case: $O(n)$

Best case: $\Omega(1)$

Sequential/linear search

numbers = [2, 95, 55, 72, 38, 46, 83, 51, 91, 17, 29]

42

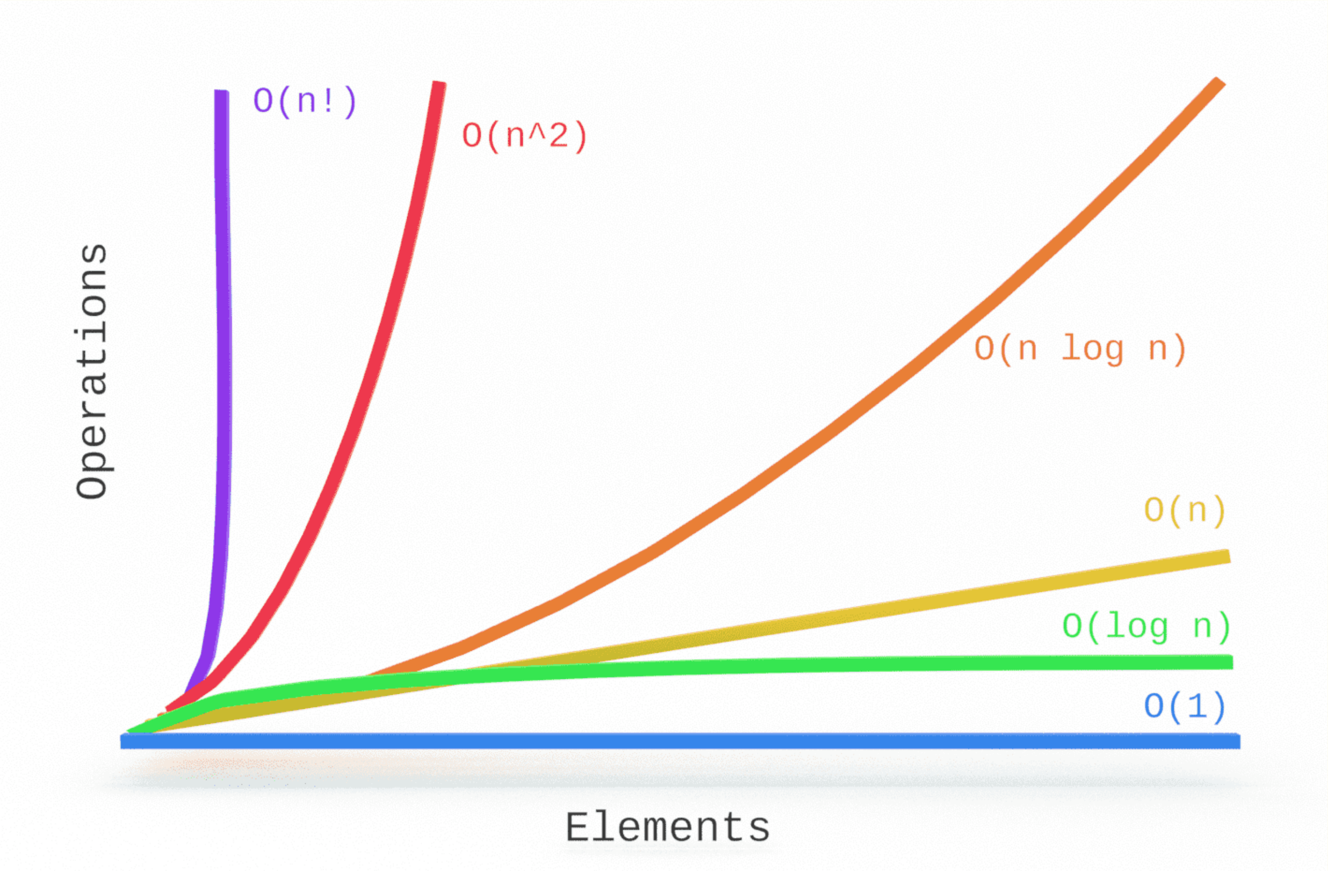
```
def sequential_search(query, items):
    for (position, item) in enumerate(items):
        if item == query:
            return position

    return -1
```

Time Complexity

Worst case: $O(n)$

Best case: $\Omega(1)$

Complexity… O(n)… what?



https://jarednielsen.com/big-o-linear-time-complexity/

How many operations do you need to perform when input size is *n?*

Asymptotic notation
Big-O notation

Constants dropped

Can we do better than sequential search?

Binary search

numbers = [2, 95, 55, 72, 38, 46, 83, 51, 91, 17, 29]

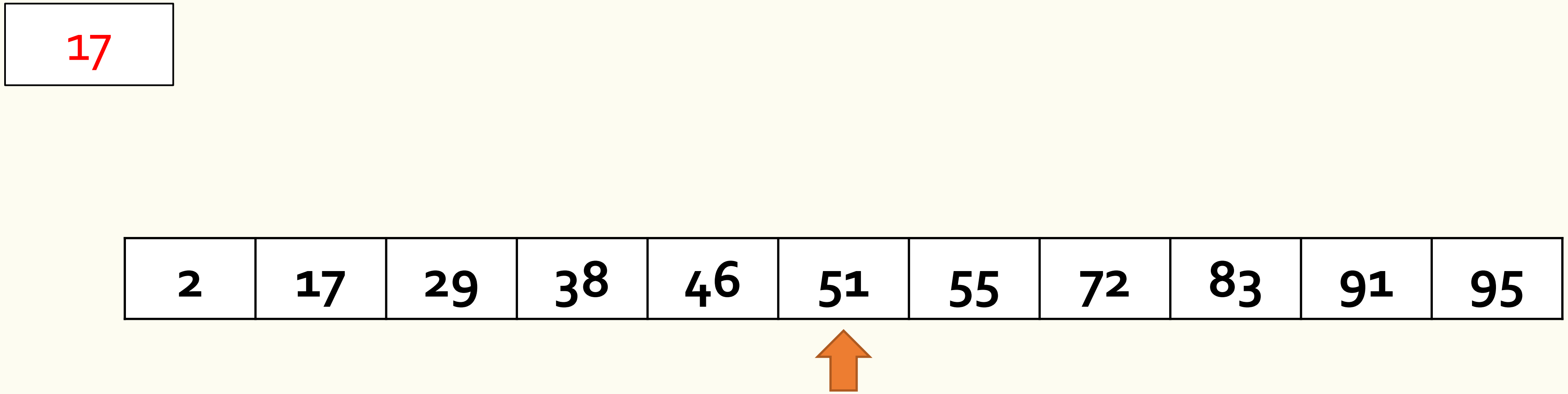
↓

numbers = [2, 17, 29, 38, 46, 51, 55, 72, 83, 91, 95]

```
import bisect, random

numbers = []
for i in range(0, 11):
    number = random.randint(1, 100)
    bisect.insort(numbers, number)
    print(numbers)
```

To keep your list always sorted

Binary search

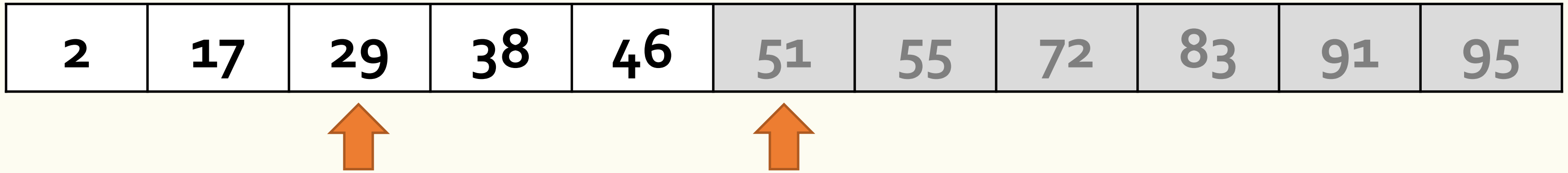


17

2	17	29	38	46	51	55	72	83	91	95

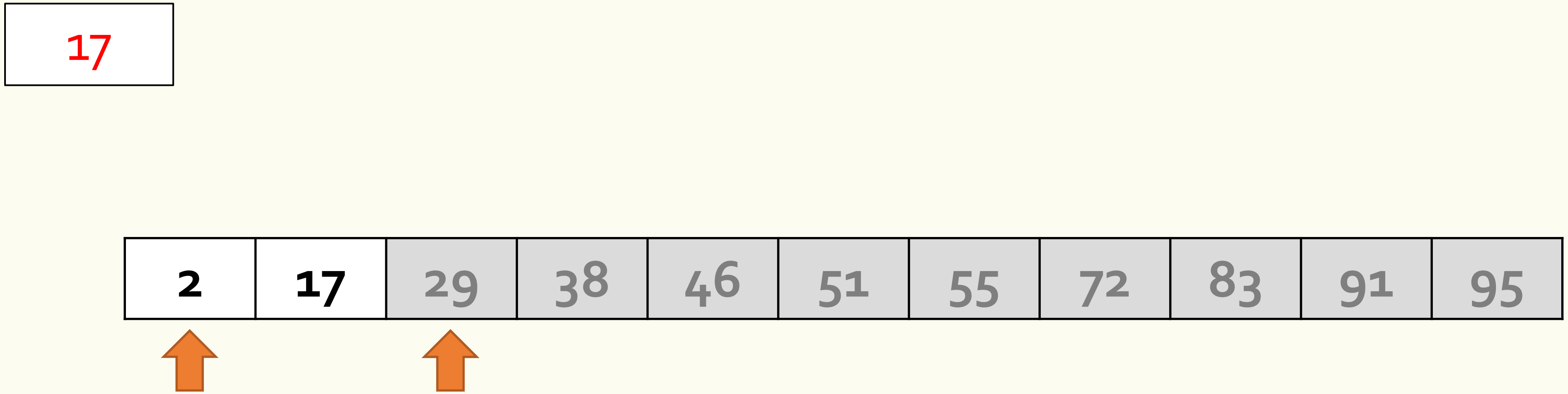
Binary search

17



2	17	29	38	46	51	55	72	83	91	95

Binary search

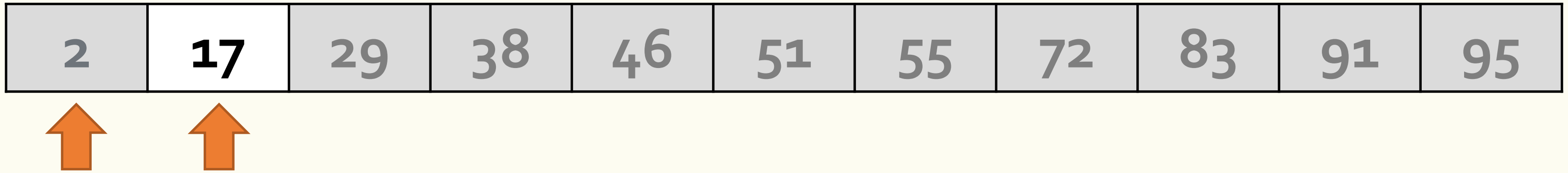


17

2	17	29	38	46	51	55	72	83	91	95

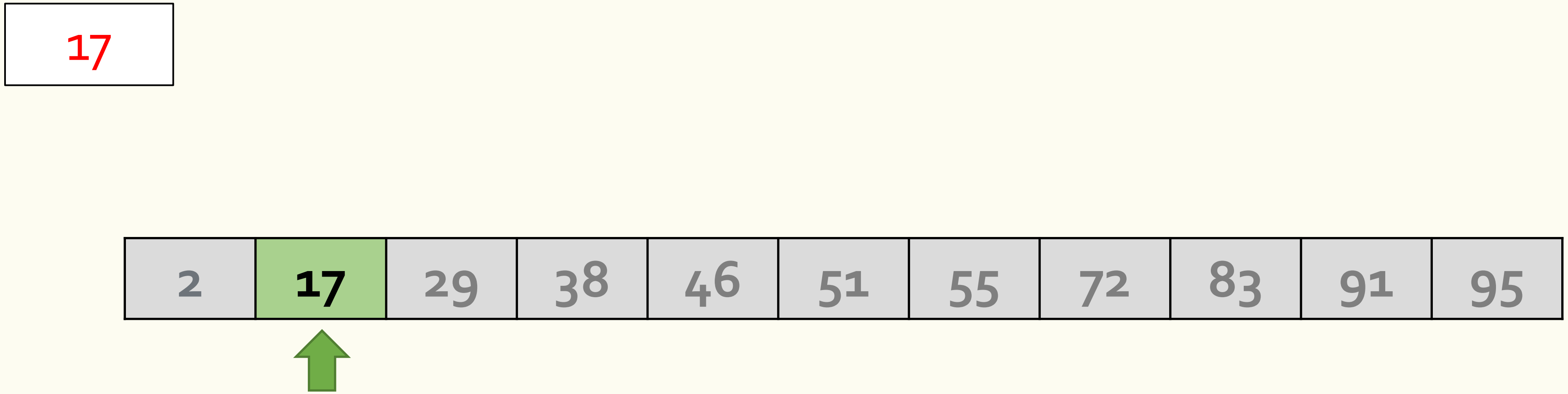
Binary search

17



2	17	29	38	46	51	55	72	83	91	95

Binary search



17

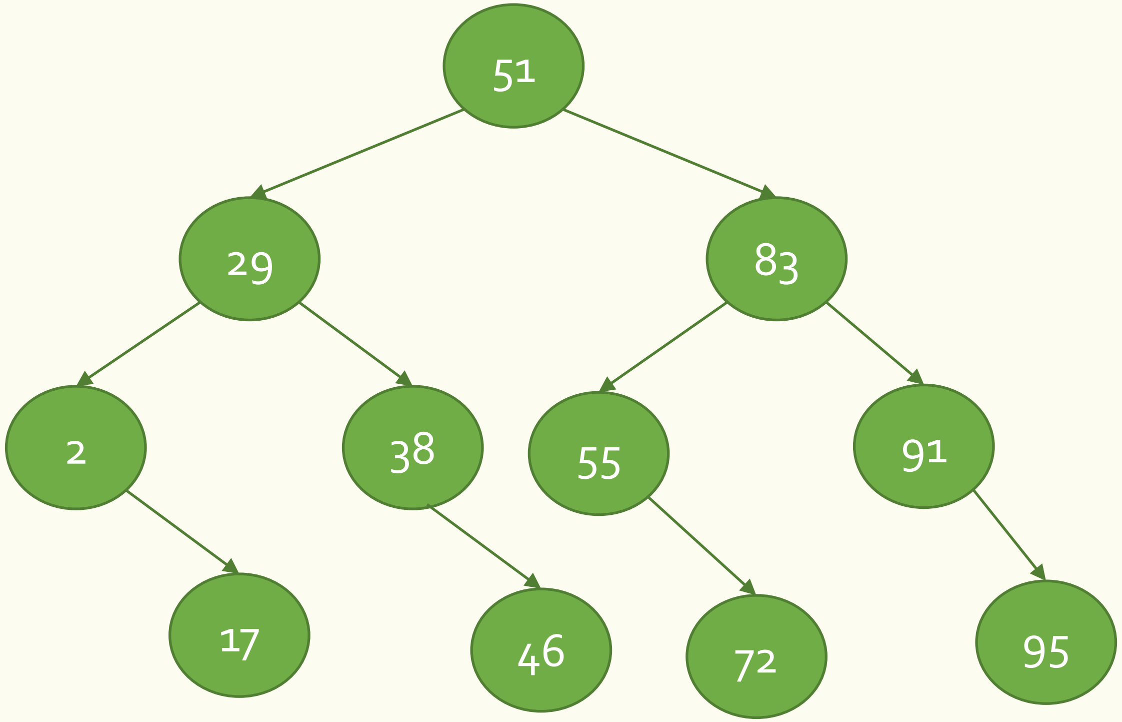
2	17	29	38	46	51	55	72	83	91	95

Time Complexity

Best case: $\Omega(1)$

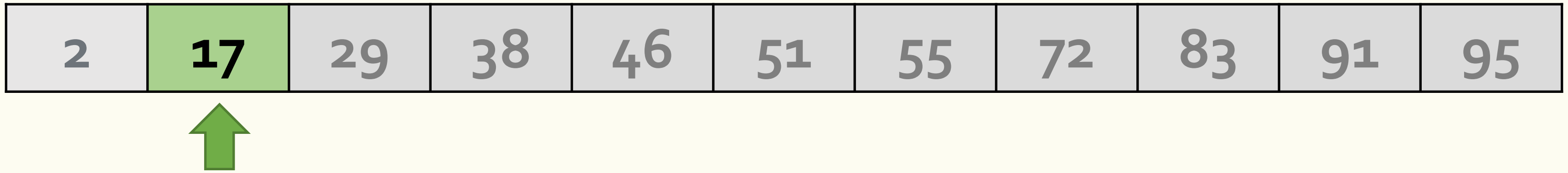
Worst case: $O(\log n)$

Binary tree



$$(\log_2 n) + 1$$

$$(\log_2 11) = 3.46$$

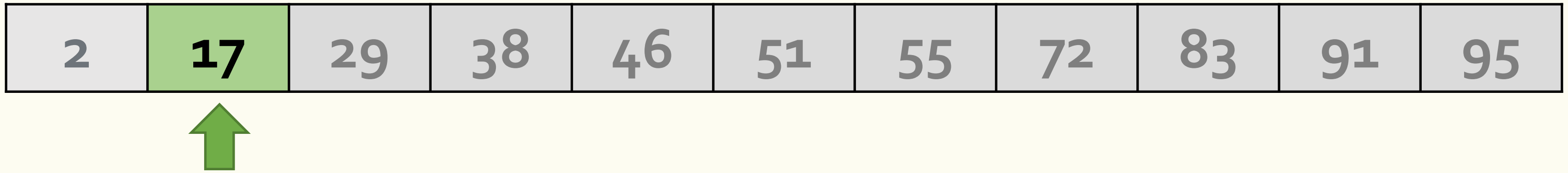


2	17	29	38	46	51	55	72	83	91	95

Implement binary search?

Iterative

Recursive



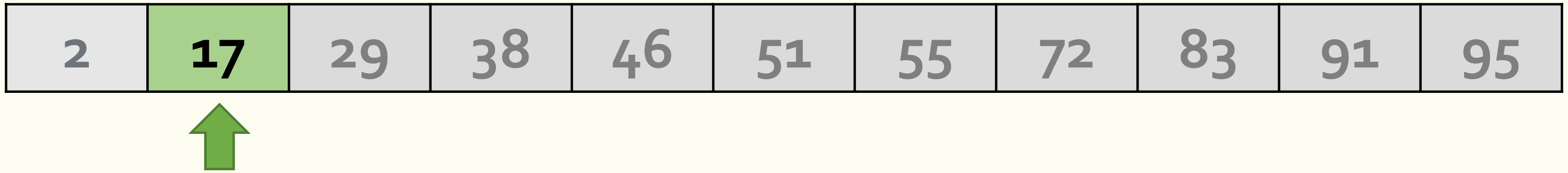
2	17	29	38	46	51	55	72	83	91	95

Implement binary search?

Iterative

Iterative implementation

```python
def binary_search(query, items):
    left = 0
    right = len(items) - 1

    while left <= right:
        middle = round(left + (right-left)/2)

        if query == items[middle]:
            return True
        elif query < items[middle]:
            right = middle - 1
        else:
            left = middle + 1

    return False
```



2	**17**	29	38	46	51	55	72	83	91	95

Implement binary search?

Recursive

Recursive implementation

```
def binary_search(query, items):
    if len(items) == 1:
        return items[0] == query
    else:
        middle = round((len(items)-1)/2)
        if items[middle] == query:
            return True
        elif query < items[middle]:
            return binary_search(query, items[:middle])
        else:
            return binary_search(query, items[middle+1:])
```

Sorting algorithms

- Bubble sort
- Insertion sort
- Selection sort
- Merge sort
- Quick sort
- Heap sort
- …

Summary

- Algorithm analysis
  - Concerned about time complexity w.r.t. input size
  - Best case, worst case
- Searching algorithms
  - Sequential search
  - Binary search
    - Iterative
    - Recursive

This week's schedule

Mon 3-4pm	Mon 4-5pm	Tue 9-10am	Wed 9-10am	Thu 11am-1pm
LECTURE Online only	LAB Online only	LAB 219	LAB 219	LAB 221/225

Next week's lecture topic: Trees

One on one with Josiah

Mon 11/10 (4PM)		
16:00-16:10	jac202	John Carter
16:10-16:20	am10118	Anagh Malik
16:20-16:30	cuo21	Chibudom Onuorah
16:30-16:40	????	Jonathan Hewlett
16:40-16:50	jh3617	Jacob Hughes-Hallett
16:50-17:00	lr4617	Lapo Rastrelli

Tue 12/10 (9AM)		
09:00-09:10	aaa1421	Abdullah Alrumayh
09:10-09:20		
09:20-09:30		
09:30-09:40		
09:40-09:50	aj2221	Alexander Jenkins
09:50-10:00	lmc16	Lucille Cazenave

One on one with Josiah

Wed 13/10 (9AM)		
09:00-09:10	av1017	Avish Vijayaraghavan
09:10-09:20	fn421	Federico Nardi
09:20-09:30		
09:30-09:40	sh2316	Simon Hanassab
09:40-09:50	cp2620	Camille Petri
09:50-10:00	atr17	Alexander Ranne

Thu 14/10 (11AM)		
11:00-11:10	cm2021	Christos Margadji
11:10-11:20	cpc21	Cormac Conway
11:20-11:30	jla21	Jonah Anton
11:30-11:40	mjc121	Matthew Collins
11:40-11:50	mt3215	Maksym Tymchenko
11:50-12:00	sk2521	Sun Jin Kim
12:00-12:10	st321	Sofiya Toteva
12:10-12:20	tap21	Thomas Phillips
12:20-12:30	yo521	Yi Siang Ong
12:30-12:40	????	Mario Lavina Martinez

Any feedback for us?

- https://www.menti.com/7qxudnnc3i
- Or go to www.menti.com and enter **1011 6313**