Algorithms and Data Structures for Data Science
lab_search

CS 277
Brad Solomon

March 10, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

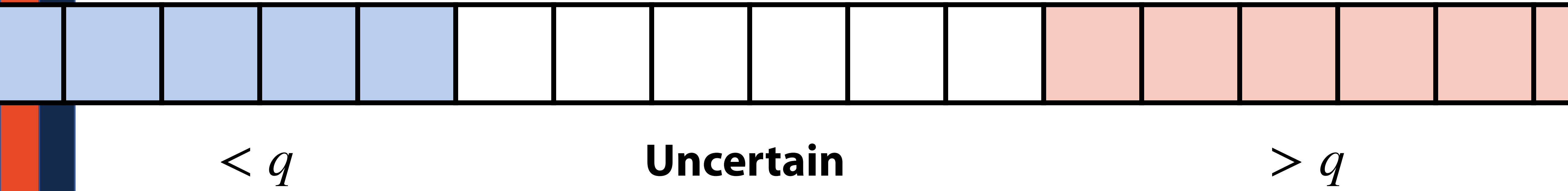
Learning Objectives

Identify limitations of binary search with multiple matches

Implement binary range searching for large-scale data analysis

Binary Search

A binary search (for object q) partitions the search space into three regions



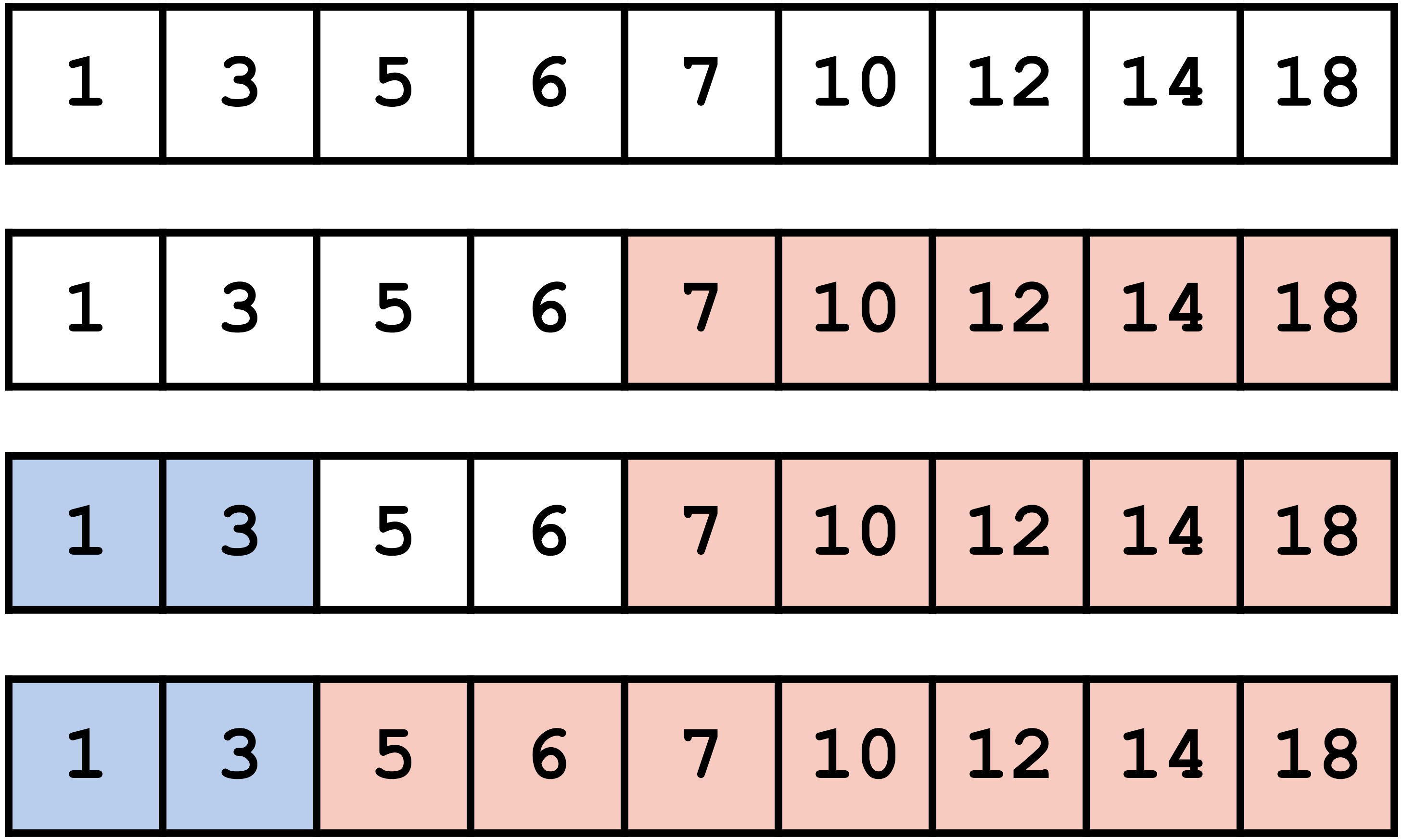
$< q$ **Uncertain** $> q$

If we are looking for q, where might we find it?

How can we track this information?

Binary Search

`Find(4)`



1 3 5 6 7 10 12 14 18

1. Find midpoint

1 3 5 6 7 10 12 14 18

2. Compare midpoint

1 3 5 6 7 10 12 14 18

3. Update range

1 3 5 6 7 10 12 14 18

Binary Search

```
def binary_search(inList, q):










def recursive_BS(inList, q, start, end):
```

0	1	2	3	4	6	7	9

Range Search

Given a collection of objects, C, with comparable values and an object of interest, q, find ~~the first~~ instance(s) of $q \in C$.

ALL

Input:

0	1	2	2	2	2	2	3	4	5

Output: Range of indices matching q if it exists, $(-1, -1)$ otherwise

Binary Range Search

`Find(3)`

Observation: All matching values are going to be consecutive

0	2	2	2	3	3	3

1. Perform binary search

2. ‘Extend’ in both directions

Binary Search: Get largest match

Find(2)

```

# THIS IS PSEUDOCODE

    if mid == q:

        # Match case:
        # Treat like query is smaller
        # Remember last match!

    elif mid > q:

        # query is smaller case
    else:

        # query is larger case

# Final Return Snippet
if saw_match:
    return last_match
else:
    return -1
```

2	2	2	2	2	2	4

Binary Search: Get smallest match

Find(3)

```

# THIS IS PSEUDOCODE

    if mid == q:

        # Match case:
        # Treat like query is larger
        # Remember last match!

    elif mid > q:

        # query is smaller case
    else:

        # query is larger case

# Final Return Snippet
if saw_match:
    return last_match
else:
    return -1


```

2	3	3	3	3	4	4

Tips

Start with binary search (and correctness)

Make sure you are hitting the efficiency benchmarks

Be careful how many times you access an item in a list!

Try range search any way you want

Then try to challenge yourself to match the optimal range search