SEARCHING

SEARCHING

- Given a collection and an element (key) to find…
- Output
  - Print a message (ex: "Found", "Not Found)
  - Return a value (position of key)
- Don't modify the collection in the search!

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

62 != 4

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

62 != 21

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

62 != 36

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

62 != 14

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

62 == 62

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

62 is at a[4]

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 4

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 21

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 36

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 14

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 62

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 91

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 8

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 22

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 7

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 81

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 77

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 != 10

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

72 is not found

UNORDERED LINEAR SEARCH

- Search an unordered array of integers for a value and return its index if the value is found. Otherwise, return -1.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
14	2	10	5	1	3	17	2

UNORDERED LINEAR SEARCH

```
Start with the first array element (index 0)
while(more elements in array){
      if value found at current index, return index;
      Try next element (increment index);
}
Value not found, return -1;
```

ALGORITHM

ORDERED LINEAR SEARCH

- Search an ordered array of integers for a value and return its index if the value is found; Otherwise, return -1.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
1	2	3	5	7	10	14	17

- Linear search can stop immediately when it has passed the possible position of the search value.

ORDERED LINEAR SEARCH

```
Start with the first array element (index 0)
while(more elements in the array){
      if value at current index is greater than value,
       value not found, return -1;
      if value found at current index, return index;
      Try next element (increment index);
}
value not found, return -1;
```

ALGORITHM

Best Case: match with the first item

Best Case:
1 comparison

target = 7

LINEAR SEARCH ANALYSIS: BEST CASE

Worst Case: match with the last item

Worst Case:
n comparisons

target = 32

LINEAR SEARCH ANALYSIS: WORST CASE

Worst Case: no match

Worst Case:
n comparisons

target = 42

LINEAR SEARCH ANALYSIS: WORST CASE

BINARY SEARCH

- Search an ordered array of integers for a value and return its index if the value is found. Otherwise, return -1.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
1	2	3	5	7	10	14	17

- Binary search skips over parts of the array if the search value cannot possibly be there.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

first **mid** **last**

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91
first					**mid**						**last**

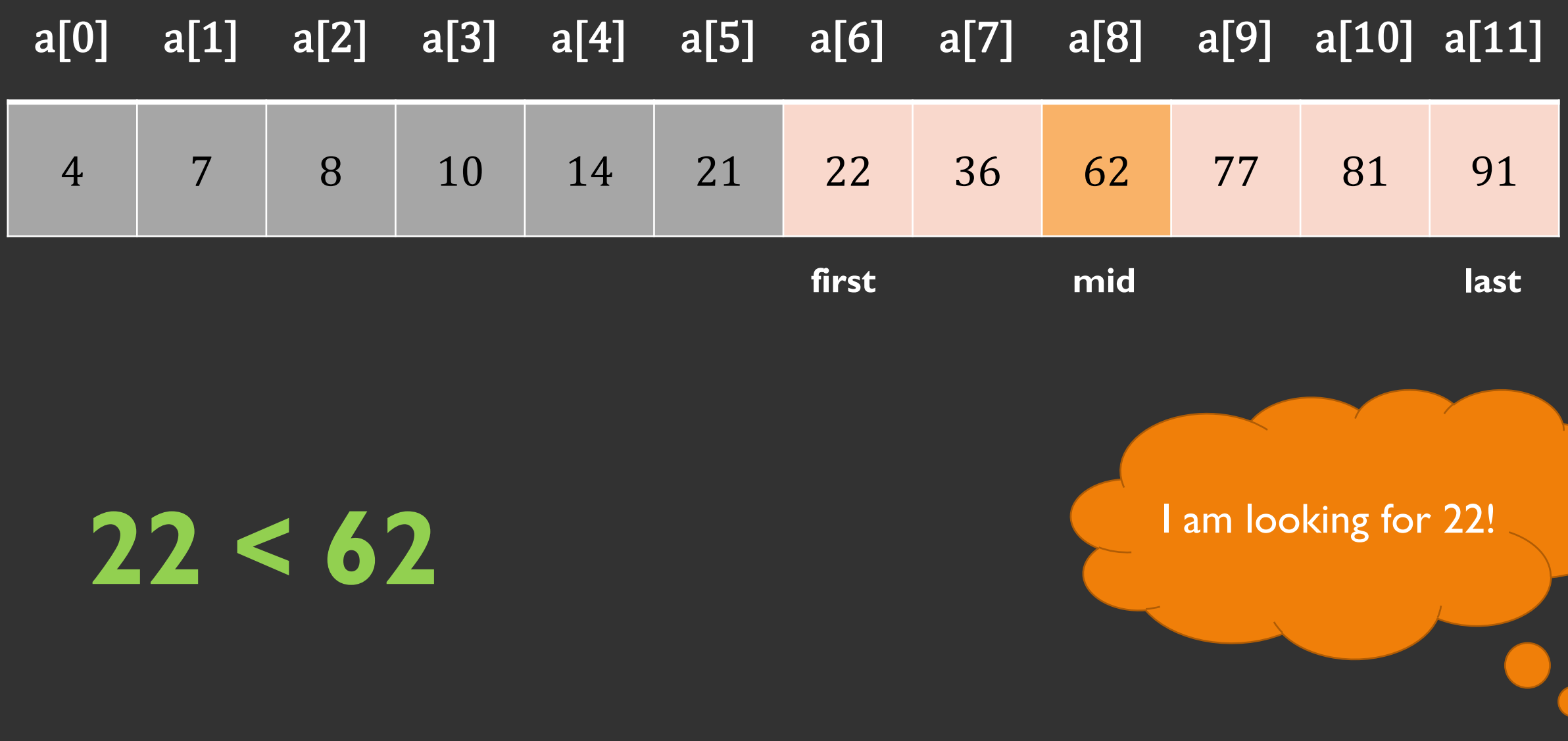
22 > 21


a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
a[10]
a[11]
4
7
8
10
14
21
22
36
62
77
81
91
first
mid
last

22 > 21


I am looking for 22!

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91
						first		**mid**			**last**


a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
a[10]
a[11]
4
7
8
10
14
21
22
36
62
77
81
91
first
mid
last
22 < 62
I am looking for 22!

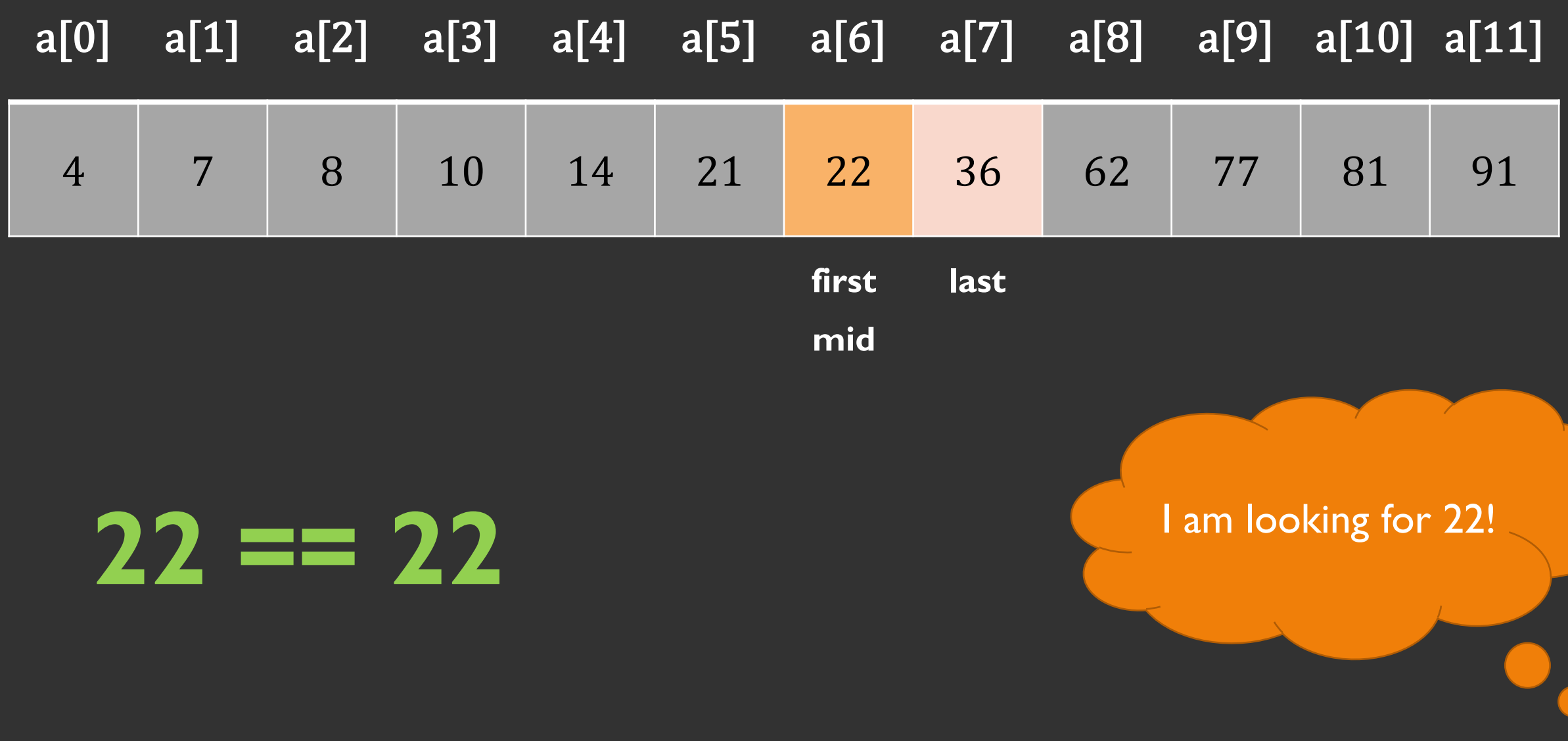
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91
						first		**mid**			**last**

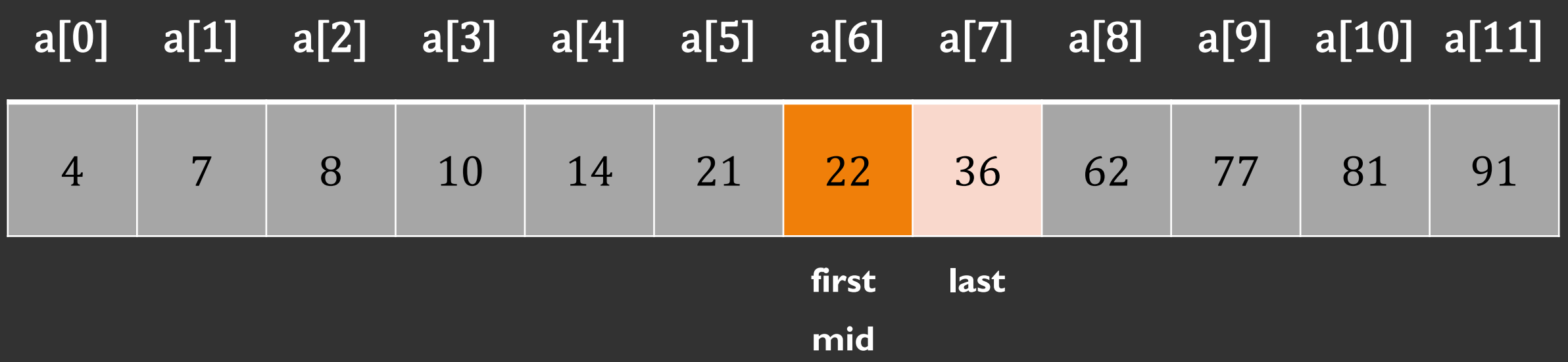
22 < 62

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

first **last**

mid


a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
a[10]
a[11]
4
7
8
10
14
21
22
36
62
77
81
91
first
last
mid
22 == 22
I am looking for 22!


a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]
a[10]
a[11]
4
7
8
10
14
21
22
36
62
77
81
91
first
last
mid

22 is at a[6]


I am looking for 22!

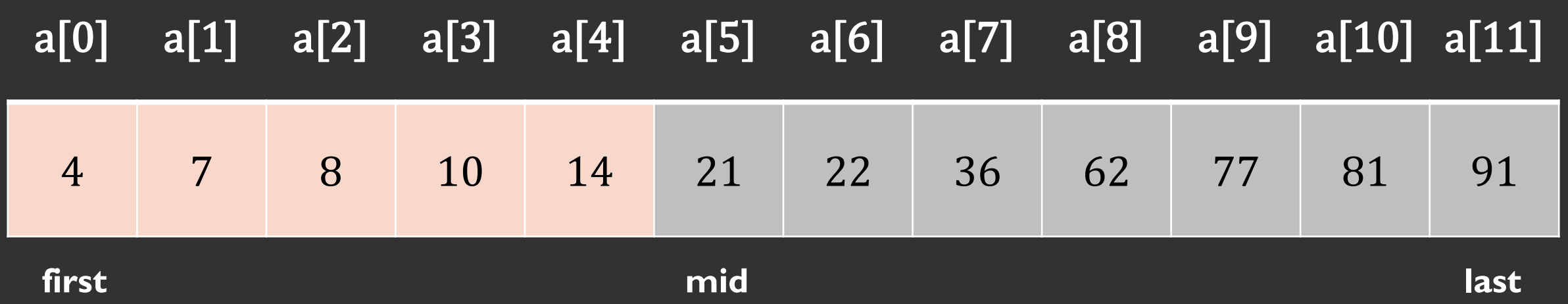
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

first **mid** **last**

11 < 21

I am looking for 11!

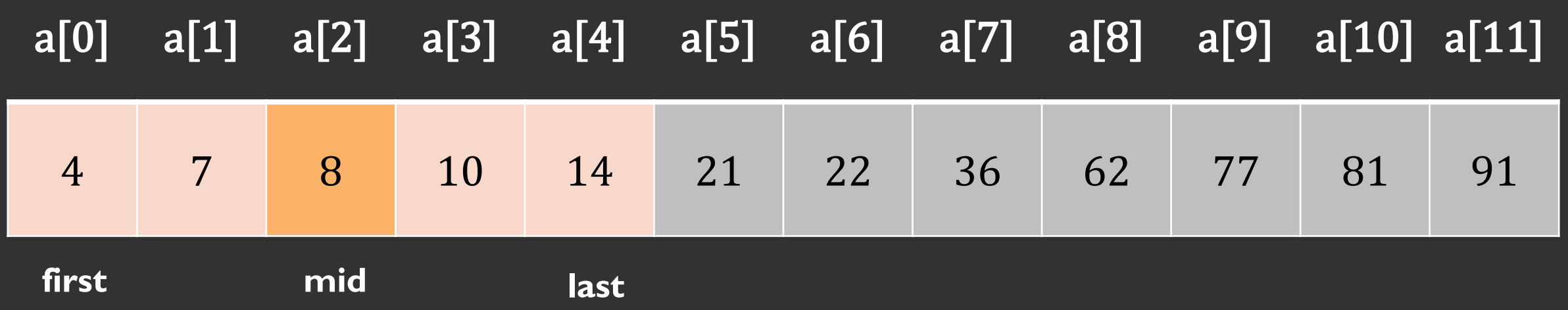


11 < 21

I am looking for 11!

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

first **mid** **last**



11 > 8

I am looking for 11!

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

first **mid** **last**

11 > 8

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

first **last**

mid

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

first **last**

mid

11 > 10

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

first **last**

mid

11 > 10

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

last

first

mid

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

last

first

mid

11 < 14

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	7	8	10	14	21	22	36	62	77	81	91

last

first

mid

11 is not found

BINARY SEARCH

- Binary search is based on the "divide-and-conquer" strategy which works as follows:
  - Start by looking at the middle element of the array
    - 1. If the value it holds is lower than the search element, eliminate the first half of the array from further consideration.
    - 2. If the value it holds is higher than the search element, eliminate the second half of the array from further consideration.
  - Repeat this process until the element is found, or until the entire array has been eliminated.

BINARY SEARCH:
A BETTER SEARCH ALGORITHM

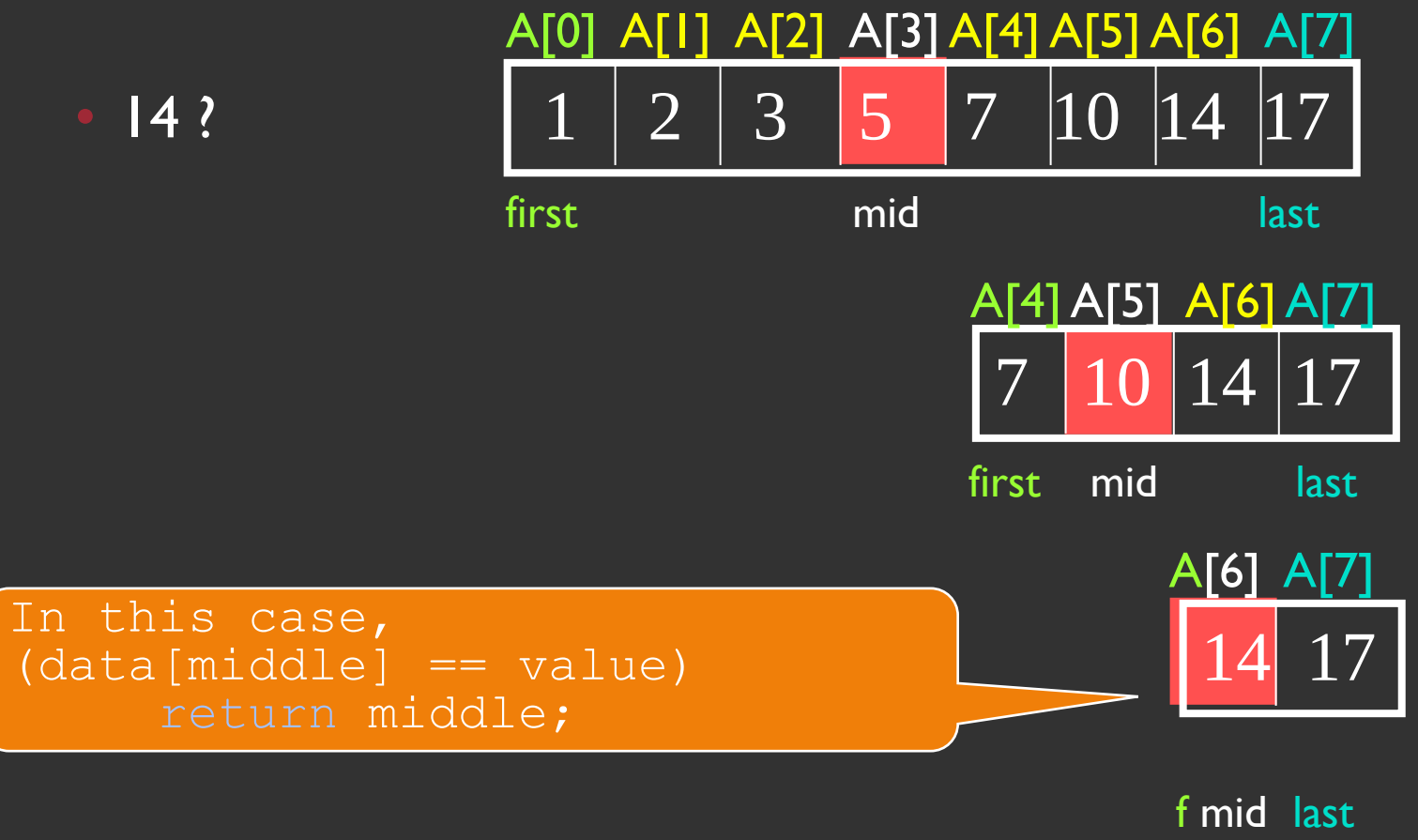
- Of course, we could use our simpler search and traverse the array
- But we can use the fact that the array is sorted to our advantage
- This will allow us to reduce the number of comparisons

BINARY SEARCH

```
Set first and last boundary of array to be searched
Repeat the following:
  Find middle element between first and last boundaries;
  if (middle element contains the search value)
          return middle_element position;
  else if (first >= last )
          return -1;
  else if (value < the value of middle_element)
          set last to middle_element position - 1;
  else
          set first to middle_element position + 1;
```

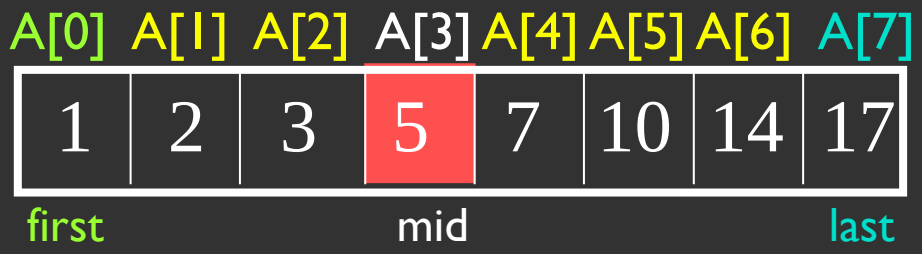
ALGORITHM

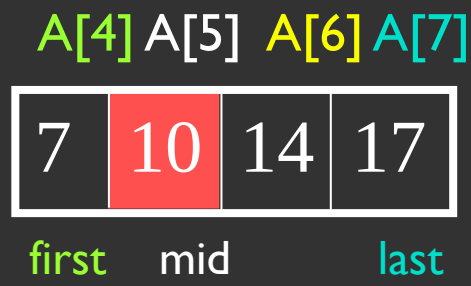
- 14 ?



BINARY SEARCH EXAMPLE

- 8 ?

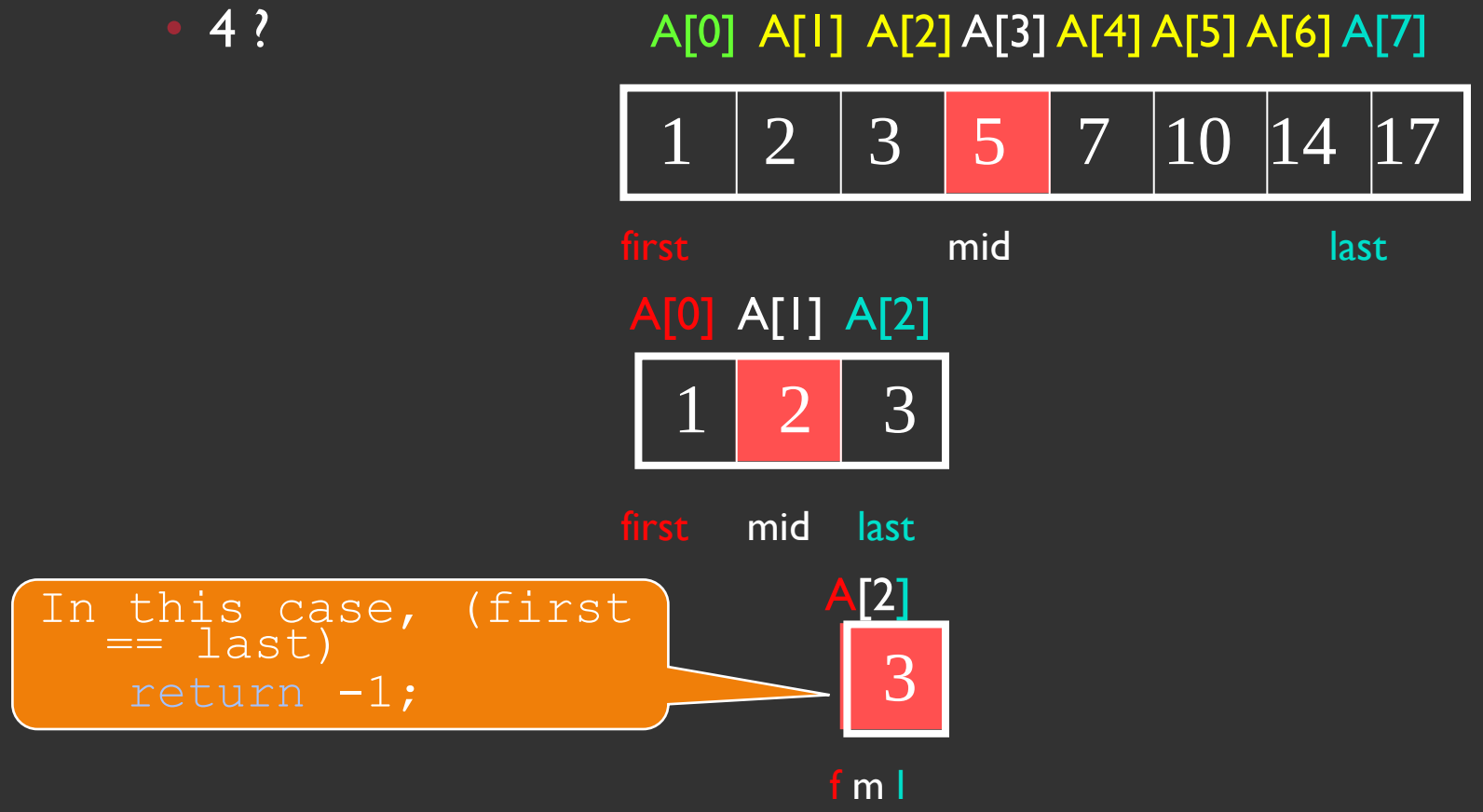





```
In this case,
   (first == last)
    return -1;
```

BINARY SEARCH EXAMPLE

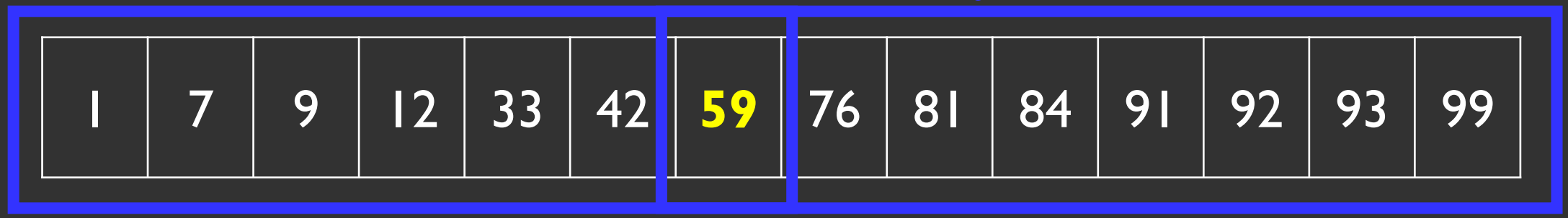
- 4 ?



BINARY SEARCH EXAMPLE

Best Case: match with the first comparison

Best Case:
1 comparison

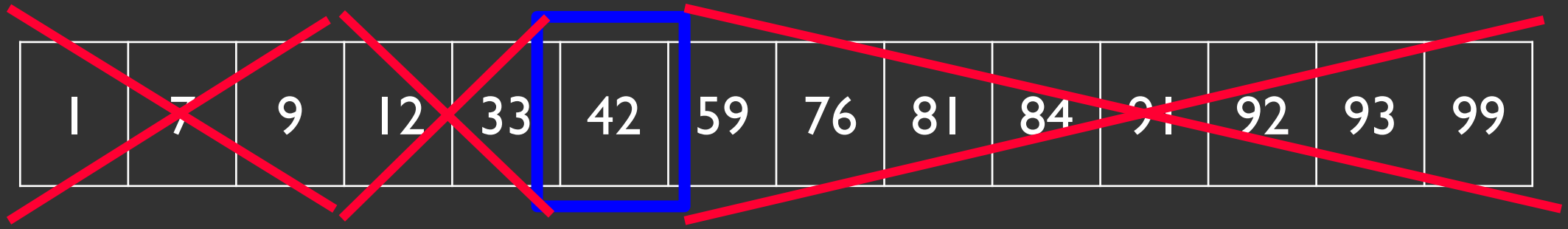


target = 59

BINARY SEARCH ANALYSIS: BEST CASE

How many comparisons??

Worst Case: divide until reach one item, or no match.



BINARY SEARCH ANALYSIS: WORST CASE

BINARY SEARCH ANALYSIS: WORST CASE

- With each comparison we throw away ½ of the list

Number of steps is at most ➔*logn*

SUMMARY

- Binary search **reduces the work by half** at each comparison
- If array is not sorted → Linear Search
  - **Best Case O(1)**
  - **Worst Case O(n)**
- If array is sorted → Binary search
  - **Best Case O(1)**
  - **Worst Case O(logn)**