

Data Structures & Algorithms

Adil M. Khan
Professor of Computer Science
Innopolis University

Searching Algorithms-1

Linear (Sequential) Search

- A very simple algorithm
 - Start at the beginning of the list
 - Move through the list, element-by-element, sequentially
 - Continue until the key is found, or
 - Until the end of the list is reached

Linear (Sequential) Search

- Time Complexity: $O(n)$
- List does not have to be sorted

Binary Search

- If the list is sorted, a more efficient search strategy can be used
- That is, check to see whether the key is
 - equal to the middle element -> terminate (found)
 - less than the middle element -> search the left half
 - greater than the middle element -> search the right half
- Continue until either the key is found, or there are no more elements to search
- Time complexity: $O(\log_2(n))$

Implementation of Binary Search in Java

```
1  /**
2   * Returns true if the target value is found in the indicated portion of the data array.
3   * This search only considers the array portion from data[low] to data[high] inclusive.
4   */
5  public static boolean binarySearch(int[ ] data, int target, int low, int high) {
6      if (low > high)
7          return false; // interval empty; no match
8      else {
9          int mid = (low + high) / 2;
10         if (target == data[mid])
11             return true; // found a match
12         else if (target < data[mid])
13             return binarySearch(data, target, low, mid - 1); // recur left of the middle
14         else
15             return binarySearch(data, target, mid + 1, high); // recur right of the middle
16     }
17 }
```

Binary Search

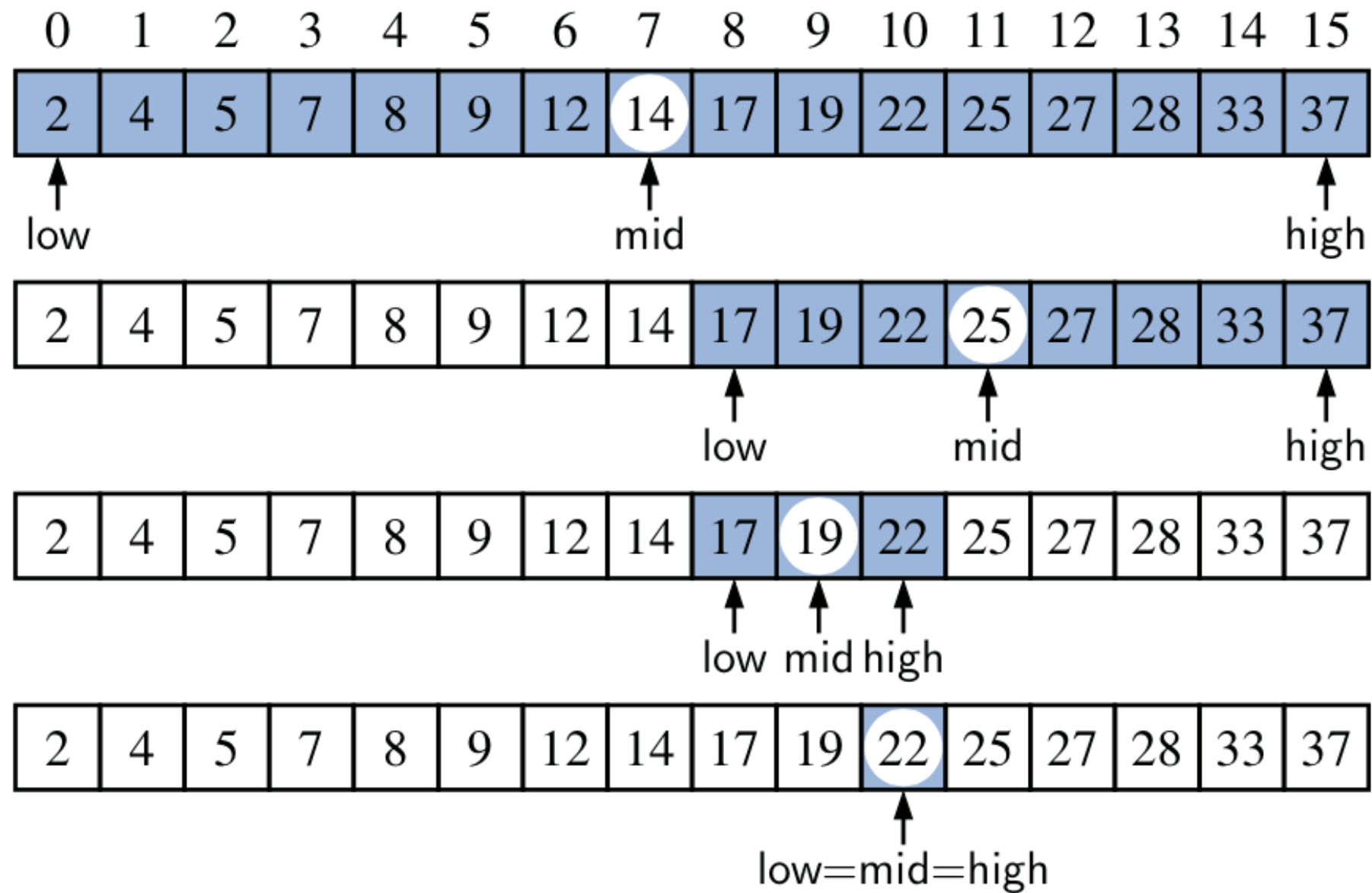
Diagram illustrating a sorted array with indices 0 to 15. The array contains the following values: 2, 4, 5, 7, 8, 9, 12, 14, 17, 19, 22, 25, 27, 28, 33, 37. Arrows indicate the positions of the first, mid, and last elements.

Index	Value
0	2
1	4
2	5
3	7
4	8
5	9
6	12
7	14
8	17
9	19
10	22
11	25
12	27
13	28
14	33
15	37

Arrows point to the first, mid, and last elements:

- first: points to index 0 (value 2)
- mid: points to index 7 (value 14)
- last: points to index 15 (value 37)

Binary Search



Example of a binary search for target value 22

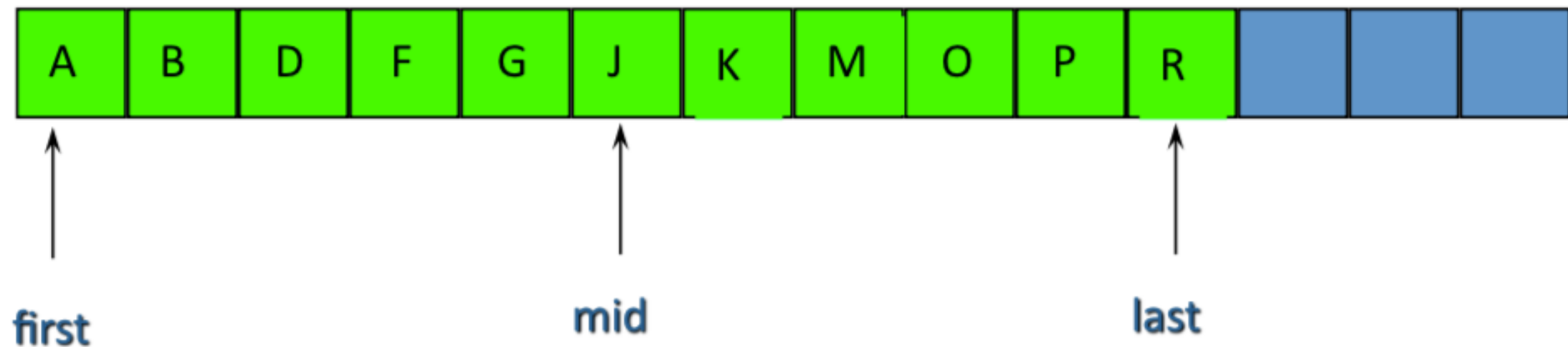
Binary Search

A	B	D	F	G	J	K	M	O	P	R			
---	---	---	---	---	---	---	---	---	---	---	--	--	--

```
first:
last:
mid:
list[mid]:
key:      P
```

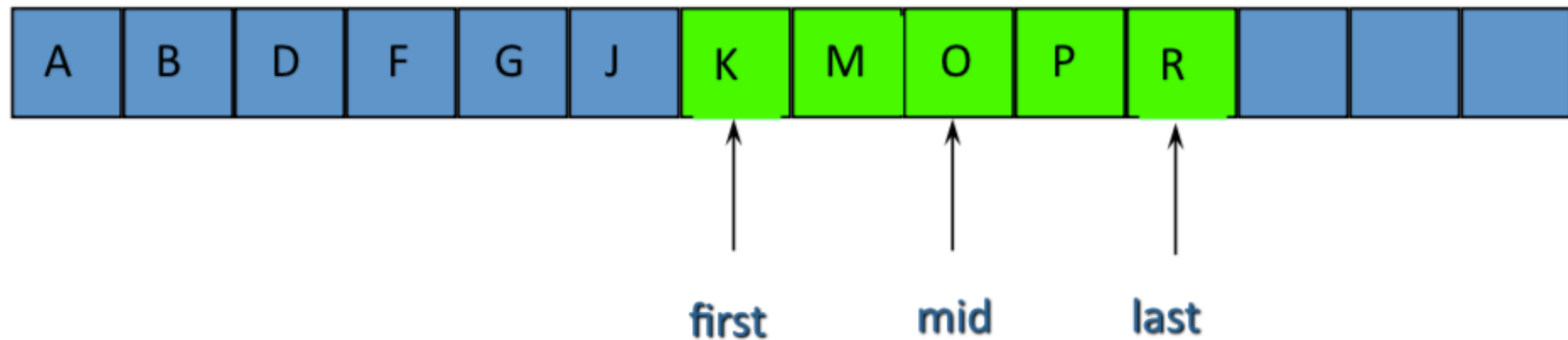
↑ ↑ ↑
first mid last

Binary Search



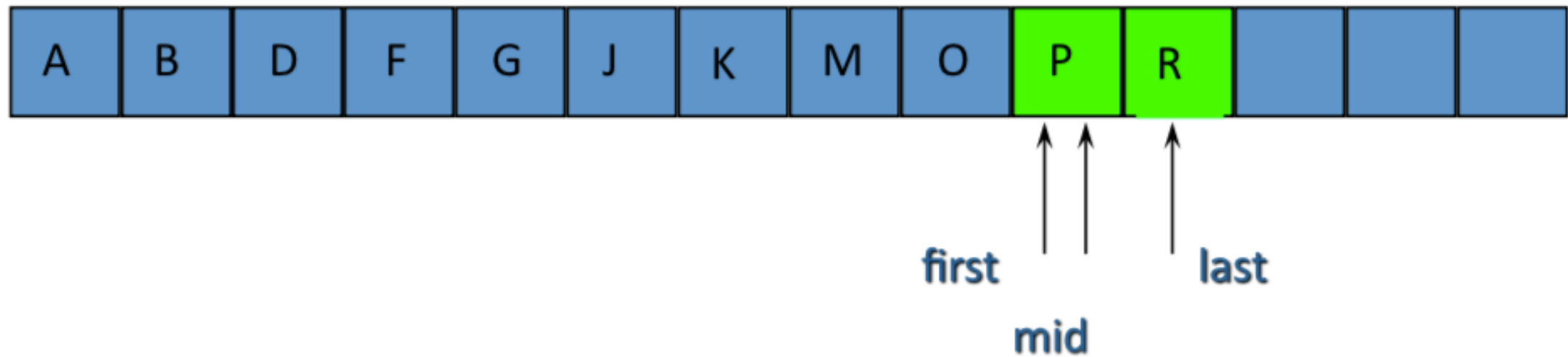
```
first:      1
last:      11
mid:        6
list[mid]:  J
key:        P
```

Binary Search



```
first:      1      7
last:      11     11
mid:        6      9
list[mid]:  J      O
key:        P      P
```

Binary Search



first:	1	7	10
last:	11	11	11
mid:	6	9	10
list[mid]:	J	O	P
key:	P	P	P

← **FOUND!**

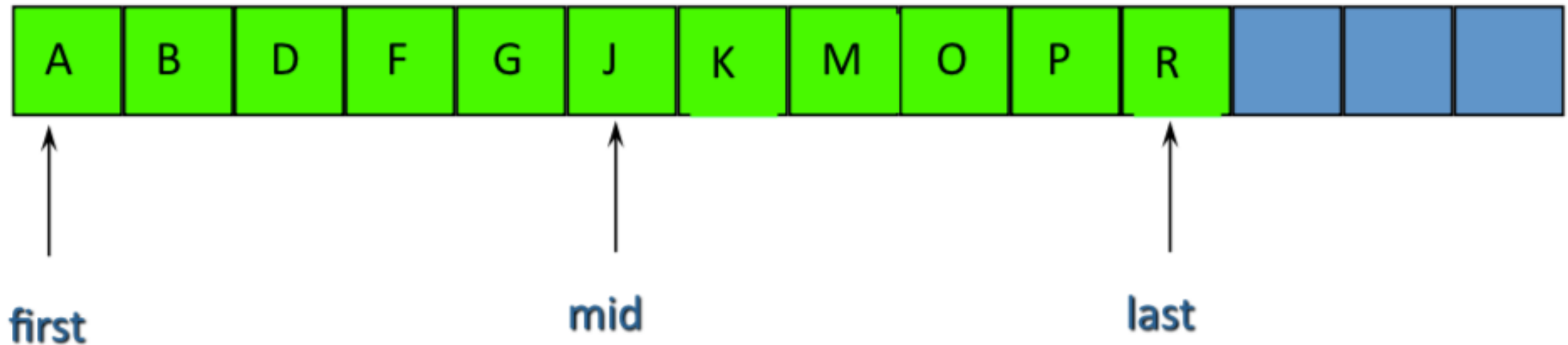
Binary Search



```
first:
last:
mid:
list[mid]:
key:      E
```

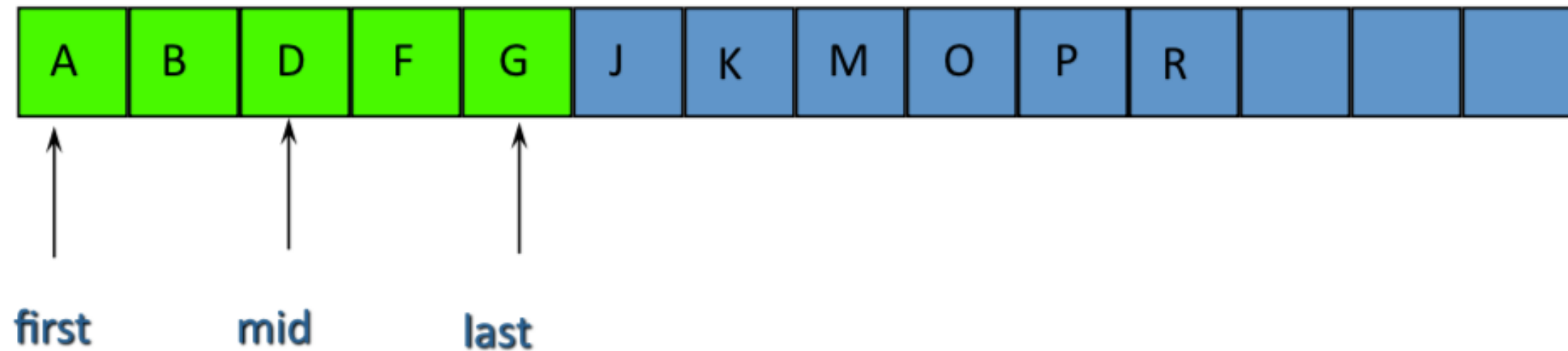
↑ ↑ ↑
first mid last

Binary Search



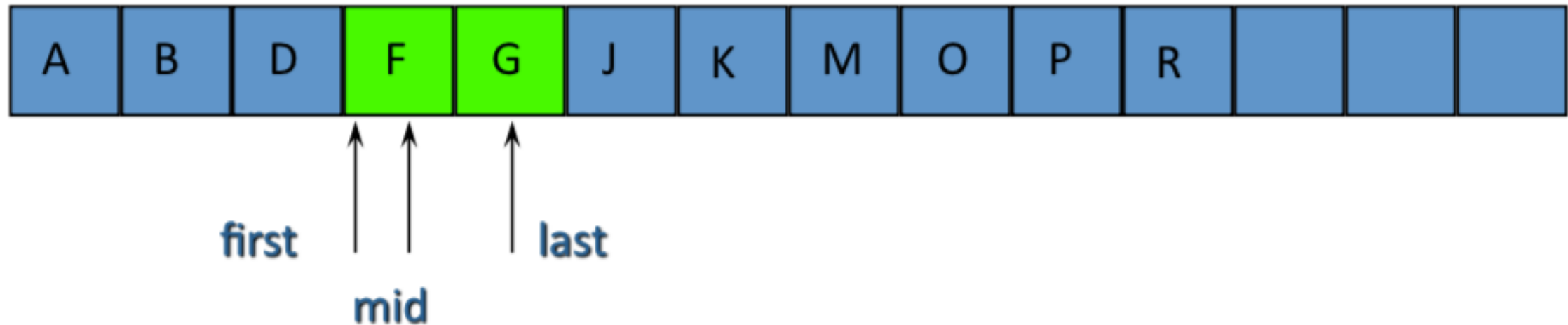
```
first:      1
last:      11
mid:        6
list[mid]:  J
key:        E
```

Binary Search



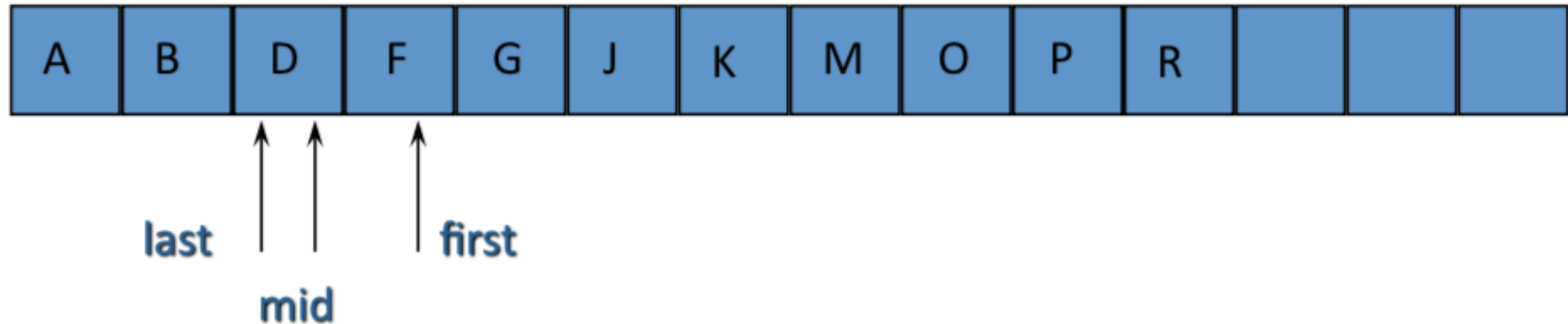
```
first:      1      1
last:      11     5
mid:        6      3
list[mid]:  J      D
key:       E      E
```

Binary Search



first:	1	1	4
last:	11	5	5
mid:	6	3	4
list[mid]:	J	D	F
key:	E	E	E

Binary Search



first:	1	1	4	4
last:	11	5	5	3
mid:	6	3	4	3
list[mid]:	J	D	F	D
key:	E	E	E	E

← first > last: NOT FOUND!