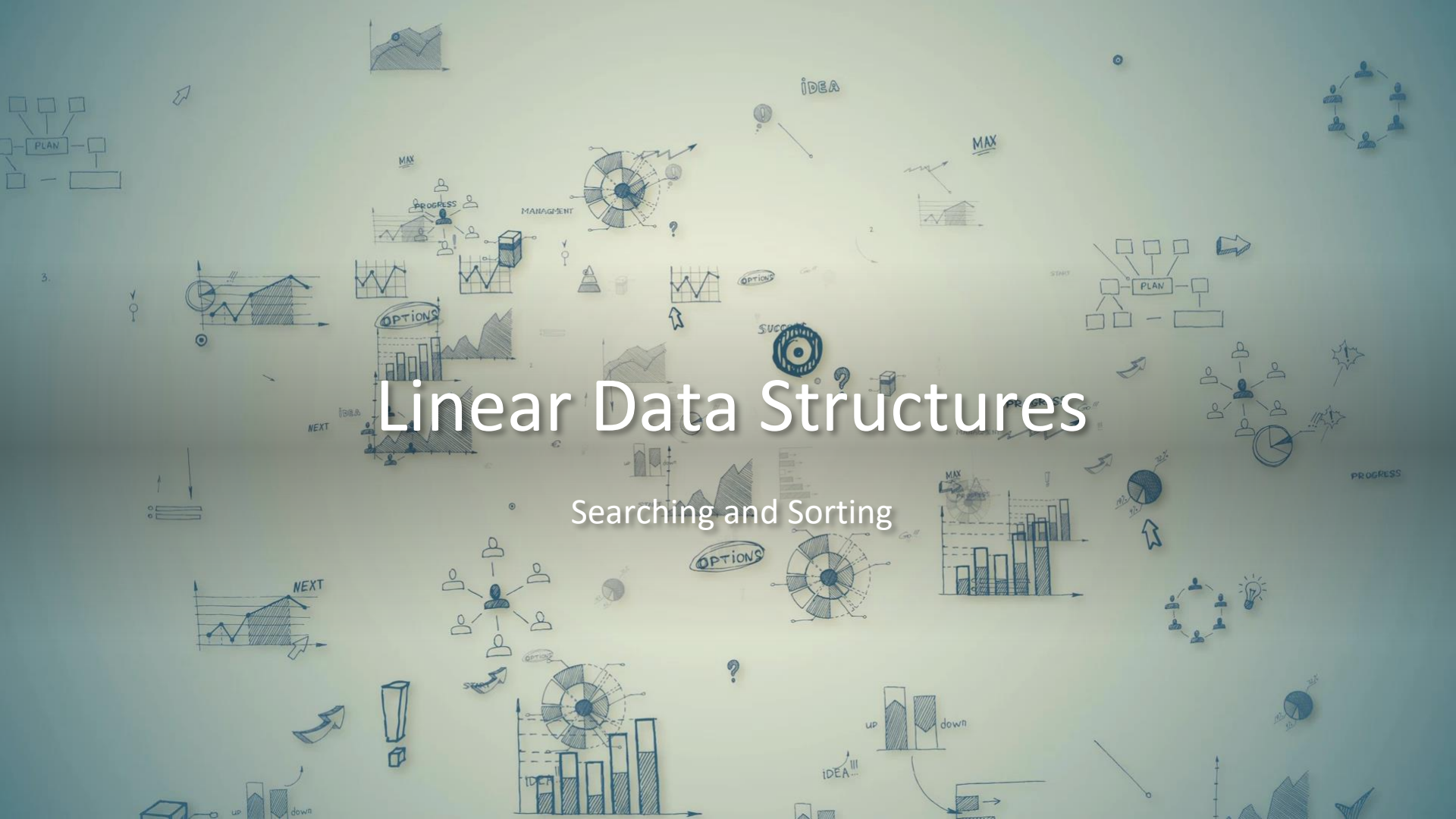


Linear Data Structures

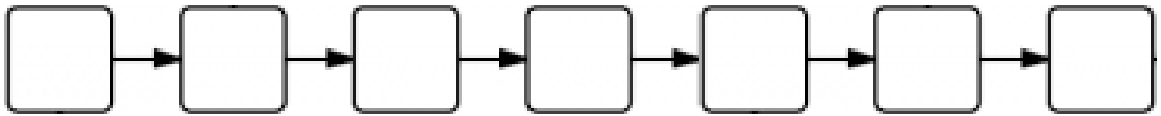
Searching and Sorting

Data Structures

- a way to store and organize data in order to facilitate access and modifications
- Categorized into:
  - Linear
  - Non-linear
- no single data structure works well for all purposes

Linear Data Structures

What is a linear data structure?
What makes a data structure “linear”?



Examples:

- Arrays
- Linked Lists
- Stacks
- Queues

Arrays

- Mainly used to store *homogenous* data
- Data is stored in consecutive memory locations
- Data is referenced by an *index*, and returns a *value*

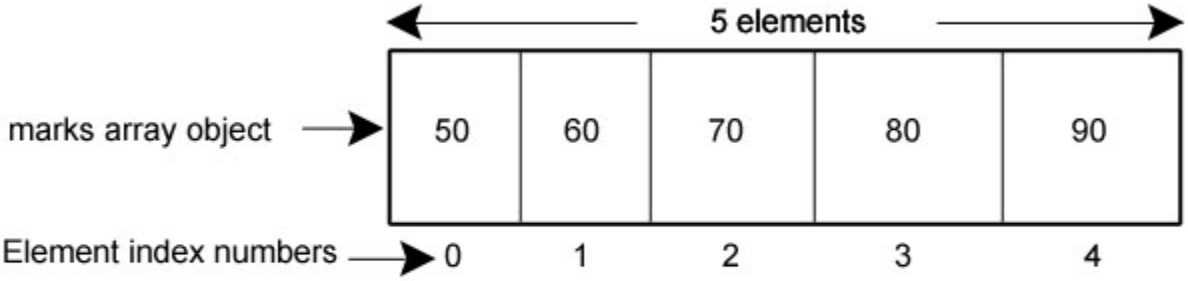
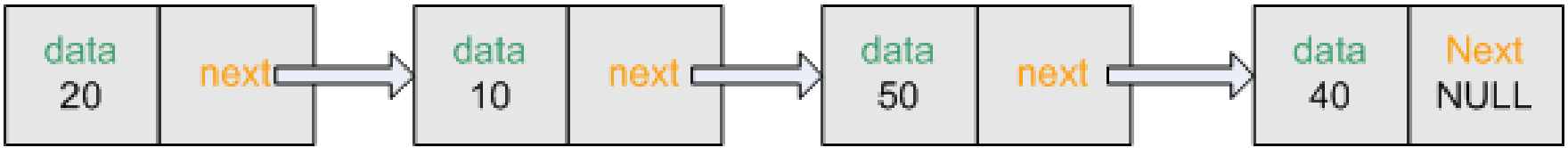


Figure : marks array with 5 elements
(after assigning values to elements)

Date	Client number	Payment
03.11.2026	24	YES
10.23.2018	30	YES
03.21.2021	6	YES
03.08.2023	24	YES
04.05.2016	29	YES
03.22.2018	10	YES
07.27.2021	30	YES
09.25.2025	5	YES
02.04.2016	23	NO
09.11.2019	20	NO
01.25.2020	28	YES
07.23.2025	18	NO
06.01.2011	25	YES
12.03.2017	14	YES

Linked Lists

- Still mainly used to store *homogenous* data
- Data is **not** stored in consecutive memory locations
- The list is **searched** until the index/element is found



Linked list

Searching

Say we have a value that we want to find in a linear data structure.

A = [6, 3, 9, 10, 54, 17, 28, 49]

How would you find 17?

Searching: Arrays

```
int arraySearch(int[] arr, int size, int val) {
        for i = 0 to size-1 {
                if (arr[i] == val)
                        return i;
        }

        return -1;
}
```

Searching: Linked Lists

```
int listSearch(list *l, int val) {
        while (*l != null) {
                if (l->val == val)
                        return l->index;
                l = l->next;
        }

        return -1;
}
```

Searching

We can see that:

- The algorithm checks, at worst case, the whole array

This algorithm is called **linear search**.

- It’s an O(n) algorithm

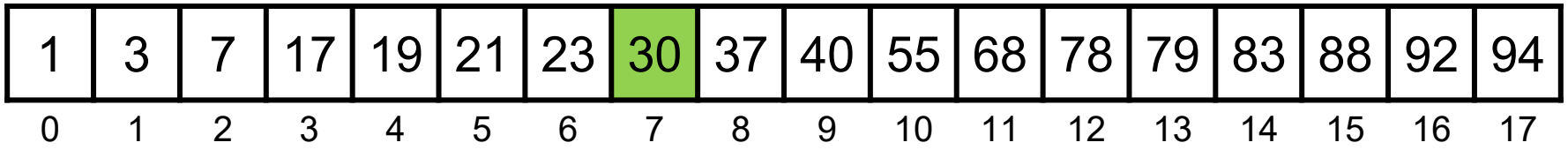
Searching

Is there a better way?

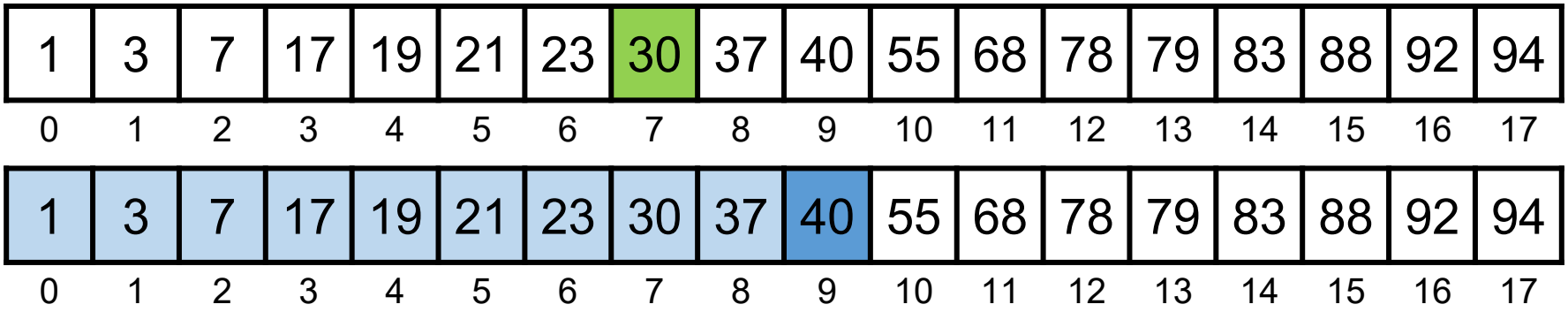
There is.

https://www.youtube.com/watch?v=wNVCJj642n4

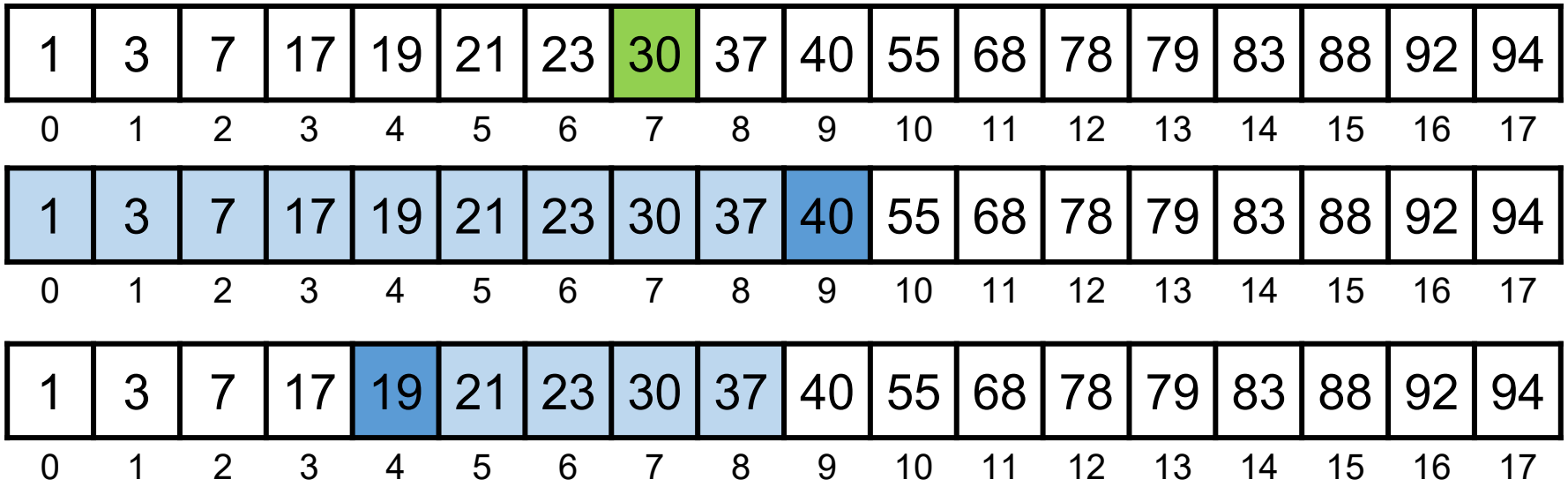
BINARY SEARCH



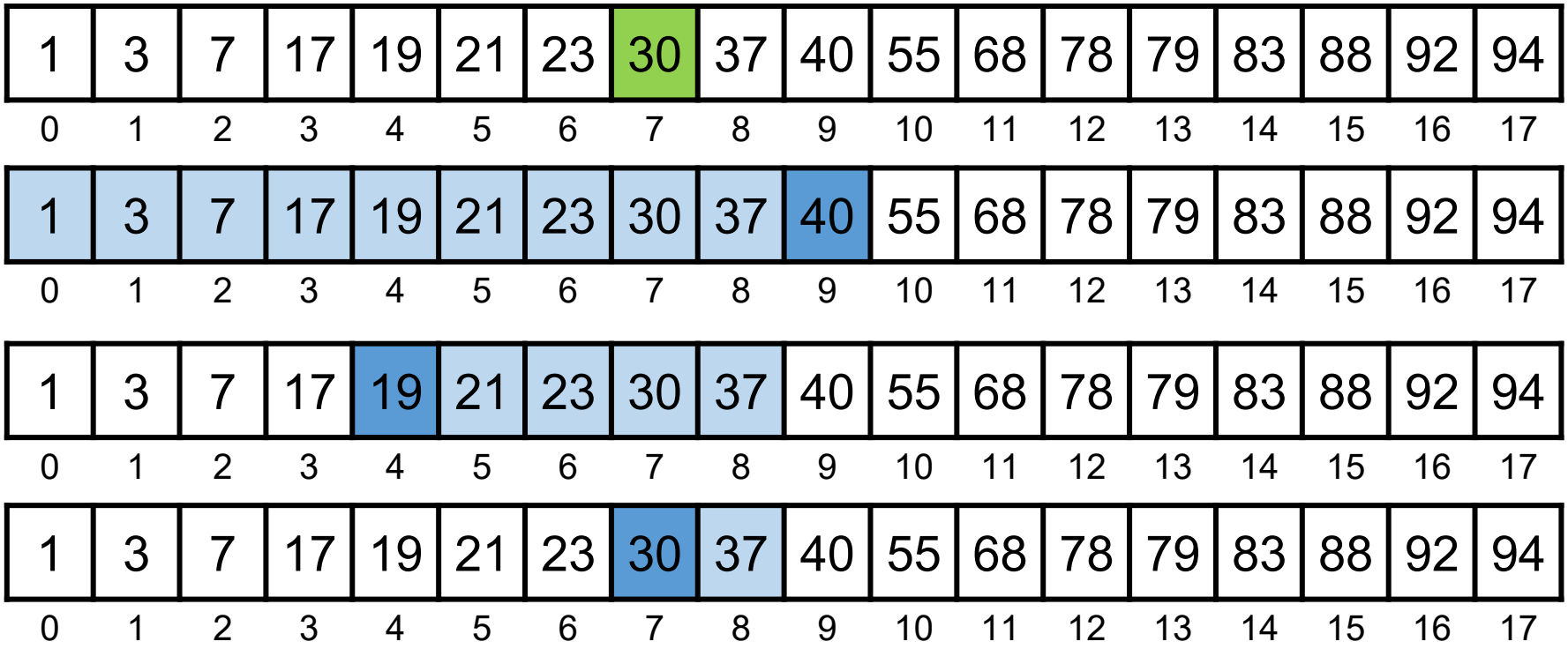
BINARY SEARCH



BINARY SEARCH



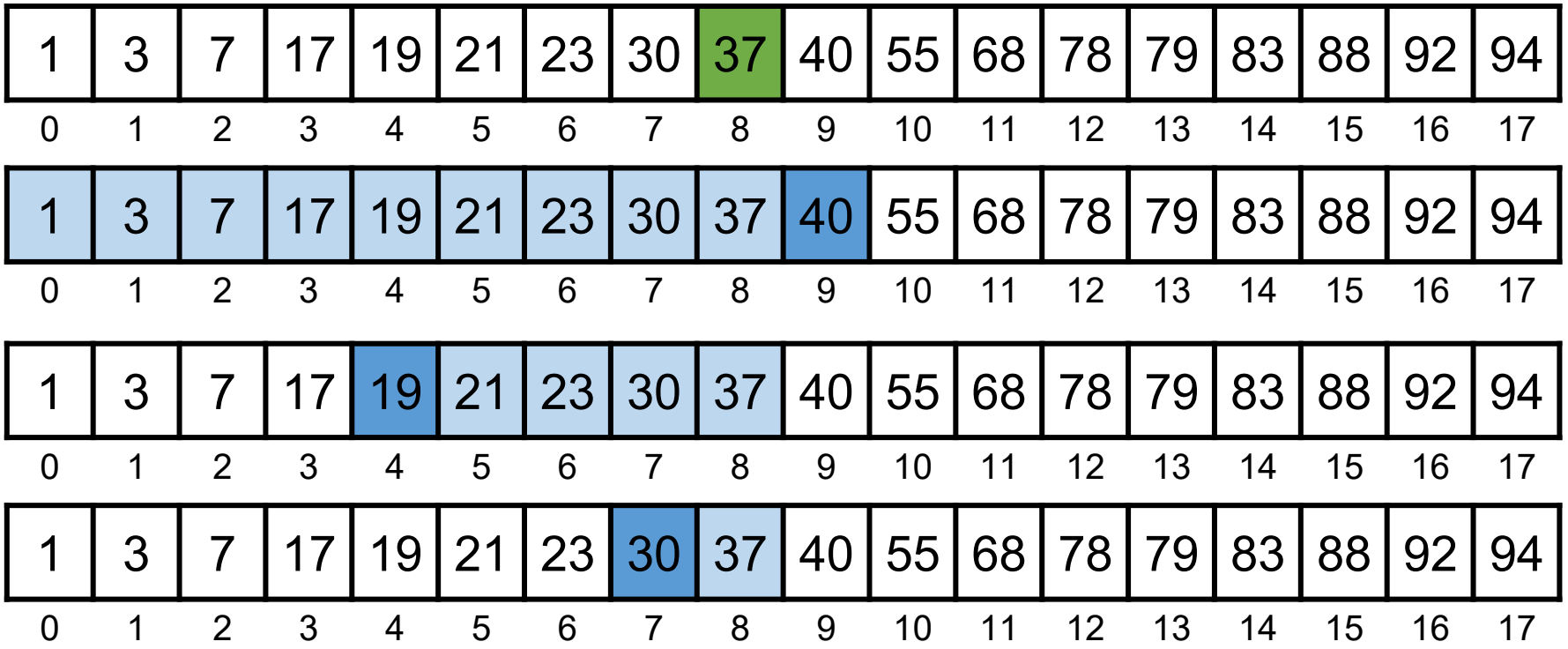
BINARY SEARCH



Searching

```
int binSearch(int[] arr, int low, int high, int x) {
    int mid;
    found = FALSE;
    while (low <= high && !found)
    {   mid = (low + high + 1)/2;
        if (A[mid] == x)
            found = TRUE;
        else if (x < A[mid])
            high = mid-1;
        else
            low = mid+1;
    }
    if (found)
        return mid;
    else /* !found */
        return -1;
}
```

BINARY SEARCH



BINARY SEARCH

```
int BinarySearch(int A[], int low, int high, int x) {
    int mid;
    if(low > high)
        return -1;
    mid = (low + high) / 2;
    if(A[mid] == x)
        return mid;
    else if(x < A[mid])
        return BinarySearch(A, low, mid-1, x);
    else
        return BinarySearch(A, mid+1, high, x);
}
```

Searching

About the algorithm:

- The algorithm now eliminates half the elements of the array for each iteration

This algorithm is called **binary search**.

- It’s an O(log n) algorithm
- **BUT! The array must be sorted first.**

Sorting: Classes

- **Internal** sorting algorithms - data is stored in an array in **main memory**
- **External** sorting algorithms - data is stored on disk or some **other device** that is best accessed sequentially

Sorting: **Properties**

In-place - no additional array storage

Stable - relative order of records with equal keys is maintained

Sorting

Say you want to sort the data in the given array:

A = [6, 3, 9, 10, 54, 17, 28, 49]

How would you sort it?

https://www.geeksforgeeks.org/sorting-algorithms/

Sorting

There are three naive approaches to sorting:

- Bubble Sort
- Selection Sort
- Insertion Sort

https://visualgo.net/en/sorting
http://bigocheatsheet.com/

They all run at $O(n^2)$

Sorting

Just like for search, we can sort faster by making some assumptions/loosening some rules.

Making use of **more space** in our sorting algorithm gives us **mergesort**.
Allowing our algorithm to **break stability** gives us **heapsort**.
Using **randomness** in our sorting algorithm gives us **quicksort**.

All of these algorithms run at O(n log n)!

Questions?