App Academy

# Arrays

## Array Metaphor

Imagine that you are a librarian who is responsible for organizing a vast collection of books.

You can imagine an array as a shelf on the bookshelf. Each slot on the shelf holds a book. Just like how you can easily find a book on specific shelf if you know what slot it is in, you can find a specific element in an array if you know its index.

In our library example, arrays would be especially useful for organizing books by categories. You could have an array for history books, science books, and so on. This would make it easy for people to find the books they need, and for you to keep track of your collection.

So, just as a librarian uses shelves and slots to organize books, programmers use arrays to organize collections of values.

# Arrays allocated in RAM

Array

| | | |
|---|---|---|
| 1 | 3 | 5 |

RAM

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Value | | | 1 | 3 | 5 | | | | |
| Address | | | $0 | $4 | $8 | | | | |

# Static Arrays

Array

| | | |
|---|---|---|
| 1 | 3 | 5 |

RAM

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Value | | | 1 | 3 | 5 | | | | |
| Address | | | $0 | $4 | $8 | | | | |

# Dynamic Arrays

Array

| | | |
|---|---|---|
| 1 | 3 | 5 |

RAM

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Value | | | 1 | 3 | 5 | | | | |
| Address | | | $0 | $4 | $8 | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Value | | | | | | | | | |
| Address | | | | | | | | | |

## Array Operations

| Operations | Big-O Time |
|---|---|
| Read/ Write ith element | |
| Insert / Remove End | |
| Insert Middle or Beginning | |
| Remove Middle or Beginning | |

## Let’s recap: Data Structure

- Arrays must be contiguous in memory
- Static arrays have a fixed size
- Dynamic arrays solve our space problem if we fill up the array with values and are the default for many languages such as Javascript or Python
- Whenever a dynamic array needs to be resized when adding new elements, it is O(n). However, the amortized time complexity is O(1).

| Operations | Big-O Time |
|---|---|
| Read/ Write ith element | O(1) |
| Insert / Remove End | O(1) |
| Insert Middle or Beginning | O(n) |
| Remove Middle or Beginning | O(n) |

# Stacks

- LIFO (last in, first out)
- Arrays are very efficient when used for stacks
- Demo: Baseball Game

| Operations | Big-O Time |
|---|---|
| Push | O(1) |
| Pop | O(1) |
| Peek | O(1) |

## When should we use a stack?

- When we want to ensure a system does not move onto another action before completing those before
- When we want to do something in reverse order
- When we want to implement an undo / redo feature
- When we want to backtrack in searching algos (e.g. path finding in a maze)
- When we use recursion; it utilizes the call stack

## Fixed Sliding Window

- Typically the sliding window has left and right pointers to represent the boundaries of the window
- We can keep track of the contents of a window using a hash table or set.
- The aim of the sliding window is to reduce the use of nested loops and instead replace it with a single loop
    - This can effectively reduce time complexity from $O(n^2)$ to $O(n)$
- This is useful when we want to find subarrays or substrings that meet a given condition within a given length
- Demo: Contains Duplicate II

## Variable Sliding Window

- Sometimes it is useful to use a variable window size instead of a fixed size
- This is useful whenever we want to the min / max sized subarray or substring
- Demo: Longest Substring Without Repeating Characters

## Two Pointers

- “two pointers” is a very large and general topic
- What is the difference between two pointer vs sliding window?
  - Very similar, and you can consider sliding window a subset of the two pointer approach
  - For a two pointer approach, we only care about the values they are pointing to but nothing in between them
- Demo: Two Sum II

Questions?

# Let’s practice!

- Review
  - Valid Parentheses
  - Merge Intervals
- After lunch
  - Container With Most Water
  - Minimum Size Subarray Sum
- Bonus
  - Asteroid Collision