Interview Prep Course

Course Overview

Purpose

This is a 4 week mini-course designed to prepare you for technical interviews focusing on fundamentals of computer science: data structures, algorithms, problem classification, and complexity analysis. We will be reviewing topics and concepts which are commonly used as the basis for questions and coding exercises in technical interviews.

We will also cover preparation strategies and tips for interviewing effectively, as you will be evaluated not just on your knowledge but on your ability to *demonstrate* that knowledge, including the critical thinking and problem solving skills that technical interviewers look for.

Program Structure

- Requires 5-10 hours a week for 4-weeks
- Meets once a week for 2-hours for a session on campus
- Bi-weekly online interview practice

Weekly Breakdown

Over the 4-weeks, the following topics will be highlighted:

- Week 1: Strings & Arrays
- Week 2: Big O Notation, Linked Lists & Hash Tables
- Week 3: Binary Trees, BFS, DFS, BST, Heaps
- Week 4: Recursion and Bit Manipulations

Topics Covered

Topics introduced are focused on practical fundamentals of software development and mobile app design including:

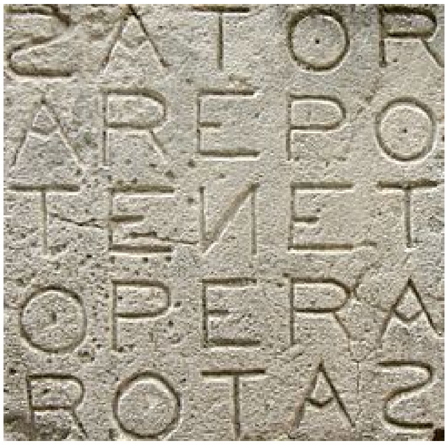
- Core data structures: Hash Tables, Arrays, Linked Lists
- Complex data structures: Binary Trees, Heaps
- Object Oriented Design/Systems Design
- Searches: Binary Search, Breadth First Search/Depth First Search
- Sorting: Merge Sort and Quick Sort
- Recursion and Combinations

Interview Prep Course

Unit 1 - Strings and Arrays

Topics

- Arrays - Ordered lists of items often used for sequential lists.
- Strings - Sequences of character data.



One of the oldest known palindromes, the Sator Square

Before the Weekly Practice Session

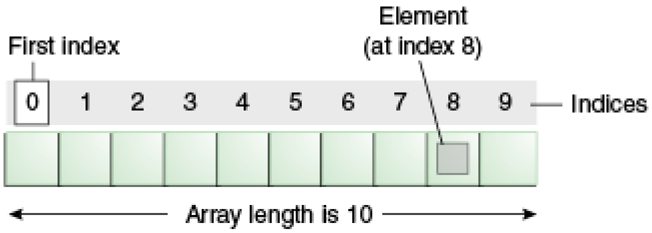
Check out the tasks before session tab to review the required tasks **before attending the weekly practice session**. Due before Self-paced.

After the Session

Check out the tasks after session tab to review the required tasks **after attending the weekly practice session**. Due before Self-paced.

Interview Prep Course

Topic: Arrays



Arrays are one of the oldest and most commonly used data structures, consisting of a sequential collection of elements, each of which is typically identified by at least one *index* value (usually starting with `0`). Index values are used to access specific elements in an array, typically via a *subscript* (example: `values[0]`). Arrays are closely related to the concept of tuples from set theory.

Types

- **One-dimensional arrays** also known as **linear arrays**, are the most basic and most common.
- **Multi-dimensional arrays** have one index per dimension. Thus a two-dimensional array has both *x* and *y* indices, and can be used to represent a grid with row and column indices, or other mathematical constructs, such as a matrix.
- **Dynamic arrays** support resizing, allowing elements to be added or removed (in Java, these are implemented as Collections, such as `java.util.ArrayList`)

Highlights

- Optimal for indexed data. Arrays work best when elements can be accessed via known index values
- Non-optimal when it comes searching, inserting, and deleting (except at the end)

Common Operations

- Traversal – iteratively accessing array elements one by one
- Insertion – adding an element at given index
- Deletion – removing an element at given index
- Searching – locating an element using given index or by value.
- Updating – modify an element at given index

Big O Efficiency

Operation	Linear	Dynamic
Indexing	*O(1)*	*O(1)*

Operation	**Linear**	**Dynamic**
Searching	*O(n)*	*O(n)*
Optimized Search	*O(log n)*	*O(log n)*
Insertion	n/a	O(n)

Interview Prep Course

Topic: Strings

A string is typically a sequence of characters, and therefore, as a data structure, a string can be thought of as a one-dimensional array of character values. However, due to the prominence of strings in information processing, strings are almost always represented in programming languages as specialized data types and can be implemented in different ways. In most languages, strings are typically implemented as arrays of *bytes* that encode character data. They may have other, language-specific qualities as well, such as termination characters. While most languages support simple conversion between strings and arrays of bytes or chars, it's useful to remember the underlying implementation of a string often differs from an array.

Highlights

Common Tasks

Big O Efficiency

Nominally equivalent to that of a one-dimensional array, though actual efficiency is highly dependent upon the underlying implementation. Additionally this is complicated by the wide usage of specialized string processing algorithms, such as Boyer-Moore with an efficiency of O(n).

Interview Prep Course

Tasks Before Session 1

Complete the following tasks **before the session** this week:

- Carefully review core concept videos and topic links for the weekly topics
- Complete 1 problem **from at least 2 category buckets** for both the Arrays and Strings problem sets. This task requires solving a minimum of 4 problems (2 for each topic).

Note: InterviewBit does not always allow for problems to be solved in every language and as a result **not all questions can be solved in Objective-C or Javascript**. If you are encountering problems, we recommend you either take questions from interviewbit and **push the solutions in your language of choice to your git repo** or you can switch to solving problems in Java. If you are switching to Java, you can use this Java syntax cheatsheet as a reference.

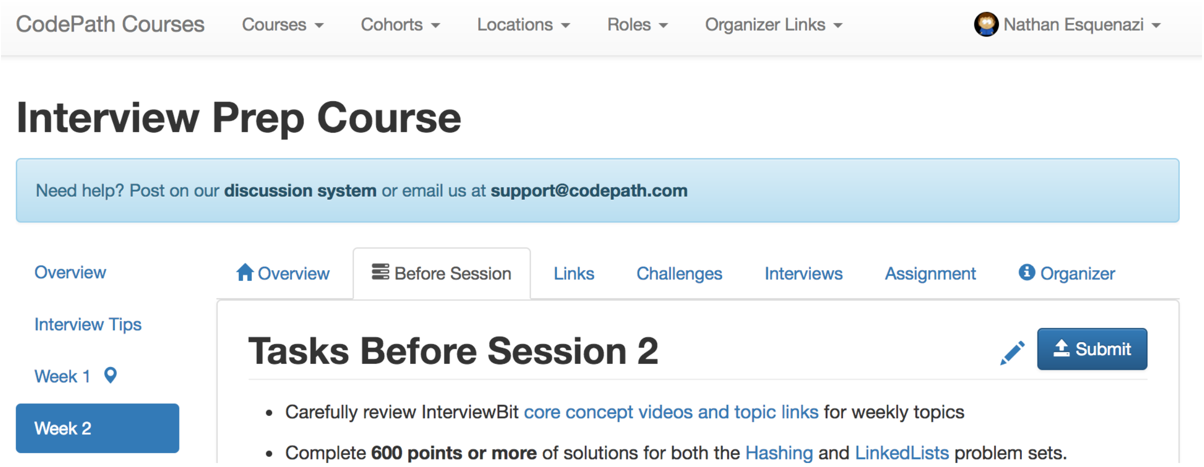
Submitting the Tasks

Once you've completed these exercises, submission involves recording a GIF of your profile / topic pages after completion.

First, go to your profile on interviewbit and record a GIF of the profile and/or topic pages that **indicates you've completed each subcategory**.

For the repository, link to your **github project repo for code solutions**. You are not required to put all your solutions into this repo, but you are encouraged to put ones you think will be most helpful to review in the future.

Next, go to the "Before Session" tab for the project and click the "Submit" button on the right-hand side:



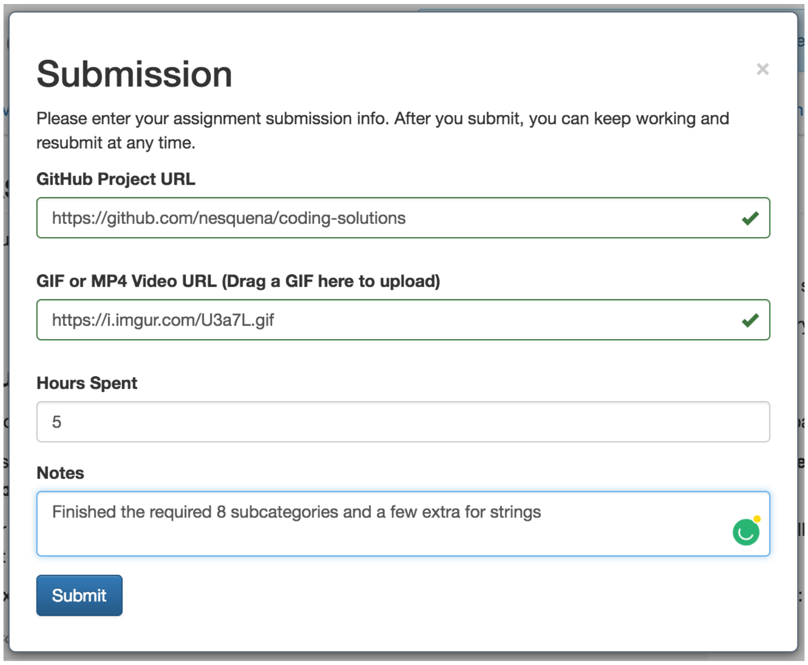
Interview Prep Course

Need help? Post on our **discussion system** or email us at **support@codepath.com**

Tasks Before Session 2

- Carefully review InterviewBit core concept videos and topic links for weekly topics
- Complete **600 points or more** of solutions for both the Hashing and LinkedLists problem sets.

In the dialog that appears, enter the required information about your project:



Submission

Please enter your assignment submission info. After you submit, you can keep working and resubmit at any time.

GitHub Project URL

https://github.com/nesquena/coding-solutions ✓

GIF or MP4 Video URL (Drag a GIF here to upload)

https://i.imgur.com/U3a7L.gif ✓

Hours Spent

5

Notes

Finished the required 8 subcategories and a few extra for strings

Then press "Submit" at the bottom to finalize your submission. Note that **you can always update your submission at any time** in case you want to re-upload the GIF or update the hours spent.

Interview Prep Course

Links - Strings and Arrays

The following resources will break down these topics in more detail. It's especially useful to check out Gayle Mcdowell's HackerRank videos in the "Core Concept Videos". She walks through how she breaks down and solves interview problems.

Strings

- Coding for Interviews Strings Guide

Core Concept Videos:

- HackerRank: Memoization
- HackerRank: Anagram Problem

Problem Sets:

- Problem Set: Strings

Additional Videos:

- Character Arrays in C++
- Character Arrays in C++ Part 2

Arrays

- Sorting Algorithms
- InterviewCake Arrays
- InterviewCake DynamicArray
- ArrayList Succinctly Guide

Core Concept Videos:

- Introduction to Sorting
- HackerRank: MergeSort

Problem Sets:

- Problem Set: Arrays

Additional Videos:

- Arrays in programming
- Pointers and 2D Array

- Printing 2D Array in Spiral Order
- Insertion Sort
- Quicksort
- Merge Sort and Analysis

Interview Prep Course

Challenges

The challenge session will **be around 45 minutes** and involves collaboratively solving the suggested problems as a group. Participants are split into groups of 3 and should do the following:

1. **Introductions** - Each person should introduce themselves to the other in the group including name, where you work, and which bootcamp you took (unless everyone has already met).
2. **Create a codepad** - Create a new collabedit document and share the URL with the other members. You'll be using this to collaboratively solve the problem.
3. **Talk through problems** - Pick any challenge and then begin to work through this problem together. Not only solving the problem but also whiteboarding the solution and discussing the solution. This isn't just about getting a correct solution but also effective communication.
   - **Tip:** Use **pseudocode rather than a particular language** when solving problems so that others in your group that prefer a different language can still understand and participate.
4. **Review solution** - After completing a problem, scroll down to the bottom of this tab to find the [interviewbit solutions repo] and discuss / compare your solution with the solution given. Is the one you've developed less efficient or less concise?

Repeat this and work through as many challenges as you can in the time alotted. At the end, if you believe you have a better solution than the one provided, submit a PR here to improve the solution set.

Problems

Here is the set of challenge problems for this week:

- Challenge 1 - Deleting duplicates from a sorted array
- Challenge 2 - Enumerate all primes <= n
- Challenge 3 - Spiral Order
- Challenge 4 - Palindrome detection
- Challenge 5 - Longest Palindromic Substring
- Challenge 6 - Longest Common Prefix

> NOTE: You are not allowed to use any libraries or specialized functions to complete the challenges below. Your responses should use pure Java, require no `import` statements, and avoid referencing external static methods or utility classes, such as `Collections` or `Arrays`.

Challenge 1 - Deleting duplicates from a sorted array

This problem is concerned with deleting repeated elements from a sorted array.

Write a program which takes as input a sorted `int[]` and updates it such that:

- all duplicates have been removed and
- all remaining valid elements have been shifted left to fill the emptied indices
- all remaining empty indices have values set to `0`
- the function returns the number of remaining valid elements (the array size minus the number of removed elements)

For example, given an input array with the values `{2,3,5,5,7,11,11,11,11,13}`, after the function completes, the values in the array should be `{2,3,5,7,11,13,0,0,0}`, and the function should return `6`.

Hint: There is an *O(n)* time and *O(1)* space solution.

Challenge 2 - Enumerate all primes <= `n`

A prime number (or a prime) is an integer greater than 1 that has no positive divisors other than 1 and itself.

Write a program which takes as input an `int` value `n` and returns an array of `int` containing all unique primes <= `n`.

Example: if the value of `n` is `8`, the function should return: `{2,3,5,7}`

Hint: One well-known algorithm for doing this is over 2,000 years old, but it's not the most efficient.

Remember, you are not allowed to use any primality testing functions.

Challenge 3 - Spiral Order

A matrix is a two-dimensional array of *r* rows, each with *c* columns, such that the total number of elements in the matrix is *r * c*.

The spiral order of such a matrix is the list of all its elements starting at index *(0, 0)* and proceeding in clockwise order from the outermost values to innermost values.

Write a program that takes an `int[][]` matrix as its input and returns an `int[]` of all the input's values in spiral order.

Example: Given the following matrix:

```
int[][] matrix = {
  { 1, 2, 3 },
  { 4, 5, 6 },
  { 7, 8, 9 }
};
```

Your program should return `{1,2,3,6,9,8,7,4,5}`

Challenge 4 - Palindrome detection

A palindrome is a word, phrase, or sequence of characters that reads the same backward as forward, e.g., *madam* or *nurses run*.

Write a program which takes a `String` as input and returns a `boolean` value which is `true` if the input is a palindrome and `false` if it is not, considering only alphanumeric characters and ignoring case.

Example:

- `"A man, a plan, a canal: Panama"` is a palindrome and should return `true`
- `"race a car"` is not a palindrome and should return `false`

Challenge 5 - Longest Palindromic Substring

Write a program which takes a `String` as input and returns a `String` which is the longest palindromic substring in the input, given the following assumptions about the input string:

- its maximum length is 1000
- it contains one unique, longest palindromic substring

Examples:

- `"abdbabbdba"` should return `"abdba"`
- `"abdbbbbdba"` should return `"abdbbbbdba"`

Challenge 6 - Longest Common Prefix

Write a program which takes a `String[]` as input and returns a `String` which is the longest common prefix, or an empty string if there is none.

Examples:

- `{"bceefgh", "bcfghijk", "bcefgh"}` should return `"bc"`
- `{"abcdefgh", "aefghijk", "abcefgh"}` should return `"a"`
- `{"", "aefghijk", "abcefgh"}` should return `""`

Solutions

- Solution for Challenge #1
- Solution for Challenge #2
- Solution for Challenge #3
- Solution for Challenge #4
- Solution for Challenge #5
- Solution for Challenge #6

Interview Prep Course

Interview Questions

The interview session will **be around 60 minutes** and involves taking turns as interviewer and interviewee. Participants are split into pairs and should do the following:

1. **Introductions** - Introduce themselves to the other person (unless they have already met).
2. **Create a Codepad** - Create a new collabedit document and share the URL with the other members. You'll be using this to solve the problem.
3. **Interview Question 1** - The person that has prepared for interview question 1 asks the question to their partner. The partner tries to understand and then solve the question for 30 minutes.
4. **Interview Question 2** - The person that has prepared for interview question 2 asks the question to their partner. The partner tries to understand and then solve the question for 30 minutes.

At the end, if you believe you have a better interview solution than the one provided (or if there is no provided solution), submit a PR here to improve our solution set.

Interview Question 1 - Gold Stars

Alice is a teacher with a class of *n* children, each of whom has been assigned a numeric rating. The classroom is seated in a circular arrangement, with Alice at the top of the circle. She has a number of gold stars to give out based on each child's rating, but with the following conditions:

- Each child must receive at least one gold star
- Any child with a higher rating than his or her immediate neighbor should get more stars than that neighbor

Assuming *n* >= 3, what is the minimum total number of stars Alice will need to give out?

Write a program which takes as its input an `int[]` containing the ratings of each child, ordered by seating position, and returns an `int` value for the minimum total number of stars that Alice will need to give out.

Hint: this problem can be solved with an algorithm that runs in *O(n)* time.

For example:

In a class of three children with ordered ratings of `{1, 2, 2}`, Alice will need to give out `{1, 2, 1}` stars accordingly, for a total number of `4` stars overall.

NOTE: You should be able to implement this in pure Java using no imports, helper functions, or collections classes!

Bonus 1

In the above example, child #3 has the same rating as child #2 but gets fewer stars. To be equitable, the number of stars should be `{1, 2, 2}`, resulting in a total number of `5` stars overall.

Modify the algorithm so that any child with fewer stars than an immediate neighbor with an equal rating gets at least as many stars as that neighbor.

Interview Question 2 - Text Justification

Given an natural language text, write an algorithm which will format the text by splitting it into lines of a specified length, with each line justified such that the text is evenly spaced according to specific rules.

Inputs:

- `String text` - the text to format
- `int length` - the line length for the output

Output: a `String[]` that meets the following requirements:

- Each line should have exactly `length` characters, including whitespace
- Each line should contain as many words as will fit, and the algorithm should return as few lines as possible
- Each line should be padded with spaces, distributed as evenly as possible
- If the number of spaces required to pad a line do not divide evenly between words, add the extra spaces between the leftmost words first
- The last line of the output should be left justified, with no extra spaces between words and all padding added to the right.

Example:

Given the inputs:

```
String text = "This is an example of text justification.";
int length = 16;
```

...the output array should be:

```
String[] output = {
  "This    is    an",
  "example  of text",
  "justification.  "
};
```

Hint: you can use `String.split(" ")` to tokenize the input into words.

Remember to avoid helper functions and libraries, though you can use methods in `java.lang.String` and classes in `java.lang` and `java.util`

Bonus 1

Add support for both left and right justification via a parameter. Use an `enum` to specify values for all three use cases, with the above us case as the default. For left and right justification, the last line should be justified the same as the rest of the text.

Using the above example inputs, the outputs for these two variants would look like this:

Left justification:

```
String[] output = {
  "This is an      ",
  "example of text ",
  "justification.  "
};
```

Right justification:

```
String[] output = {
  "      This is an",
  " example of text",
  "  justification."
};
```

Interview Prep Course

Assignment 1

You are required to complete a few additional items this week after the session:

- **Required:** Complete 1 problem **from at least 2 additional category buckets** for both the Arrays and Strings problem sets. This requires you've completed a **total of 1 problem for 8 category buckets** (4 per topic).
- **Optional:** Complete 4 more problems of your choosing between the Arrays and Strings problem sets.

Note: InterviewBit does not always allow for problems to be solved in every language and as a result **not all questions can be solved in Objective-C or Javascript**. If you are encountering problems, we recommend you either take questions from interviewbit and **push the solutions in your language of choice to your git repo** or you can switch to solving problems in Java. If you are switching to Java, you can use this Java syntax cheatsheet as a reference.

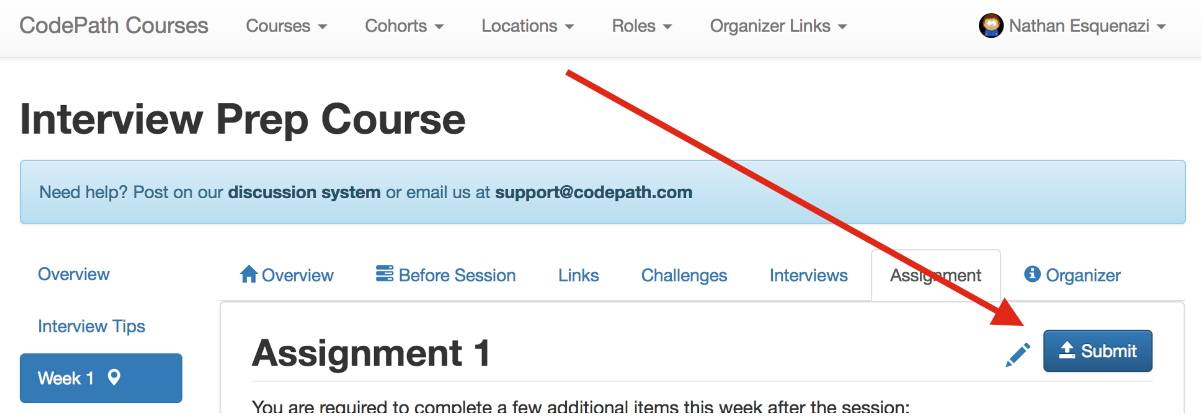
Submitting the Assignment

Once you've completed this assignment, submission involves recording a GIF of your profile / topic pages after completion.

First, go to your profile on interviewbit and record a GIF of the profile and/or topic pages that **indicates you've completed each subcategory**.

For the repository, link to your **github project repo for code solutions**. You are not required to put all your solutions into this repo, but you are encouraged to put ones you think will be most helpful to review in the future.

Next, go to the "Assignment" tab for the project and click the "Submit" button on the right-hand side:



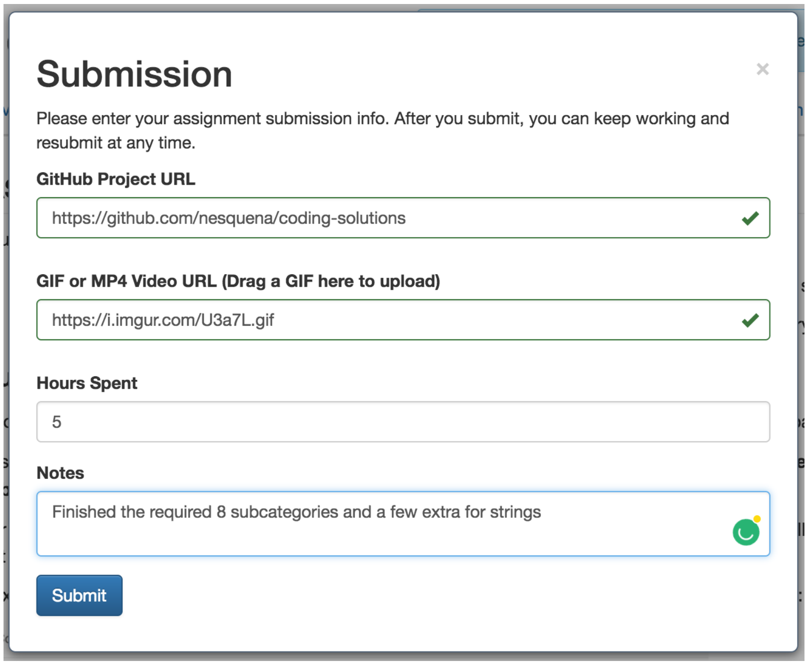
Interview Prep Course

Need help? Post on our **discussion system** or email us at **support@codepath.com**

Assignment 1

You are required to complete a few additional items this week after the session:

In the dialog that appears, enter the required information about your project:



Submission

Please enter your assignment submission info. After you submit, you can keep working and resubmit at any time.

GitHub Project URL

https://github.com/nesquena/coding-solutions ✓

GIF or MP4 Video URL (Drag a GIF here to upload)

https://i.imgur.com/U3a7L.gif ✓

Hours Spent

5

Notes

Finished the required 8 subcategories and a few extra for strings

Then press "Submit" at the bottom to finalize your submission. Note that **you can always update your submission at any time** in case you want to re-upload the GIF or update the hours spent.