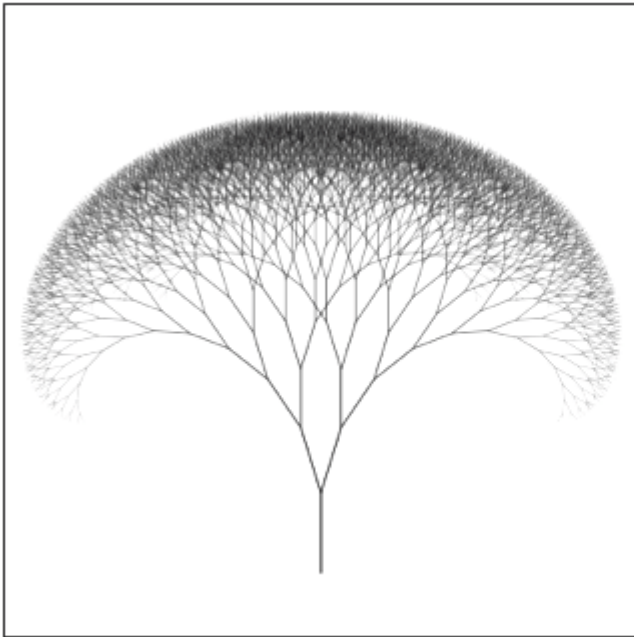


# Interview Prep Course

## Unit 4 - Recursion and Bit Manipulation

---



```
var ctx = canvas.getContext('2d'),
    generation = 0,
    branchLength = 60;

ctx.translate(200, 360);
branch(0);

function branch(angle) {

    generation++;

    ctx.save();

    ctx.rotate(angle);
    ctx.scale(0.85, 0.85);

    ctx.beginPath();
    ctx.moveTo(0, 0);
    ctx.translate(0, -branchLength);
    ctx.lineTo(0, 0);
    ctx.stroke();

    if(generation <= 12) {
        branch( 0.3);
        branch(-0.3);
    }

    ctx.restore();

    generation--;
}
```

*A simple recursive tree generation, along with the code used to create it (source)*

### Topics

- Recursion - A method where the solution to a problem depends on solutions to smaller instances of the same problem
- Bit Manipulation - At a deep level, your computer works with bits— 1s and 0s. Sometimes we want to work directly with bits using bitwise operations.

### 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**.

# Interview Prep Course

## Tasks Before Session 4

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 Recursion and Bit Manipulation problem sets. This task requires solving a minimum of 4 problems (2 for each topic).
- Prepare for your mock interview session by reviewing one of the questions and being prepared to interview another person by asking them this question. **Note:** If your birthday **day is an odd number**, prepare to ask someone question #1, otherwise if your **birthday day is an even number** then prepare to ask someone question #2.

**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 off of interviewbit and instead **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:

[CodePath Courses](#) [Courses ▾](#) [Cohorts ▾](#) [Locations ▾](#) [Roles ▾](#) [Organizer Links ▾](#)  Nathan Esquenazi ▾

## Interview Prep Course

Need help? Post on our [discussion system](#) or email us at [support@codepath.com](mailto:support@codepath.com)


[Overview](#) [Before Session](#) [Links](#) [Challenges](#) [Interviews](#) [Assignment](#) [Organizer](#)

[Interview Tips](#)

[Week 1](#) [Week 2](#)

### 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.

 [Submit](#)

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 

Submit

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 - Recursion and Bit Manipulation

---

### Recursion

- [TopCoder Recursion Guide](#)
- [Programming Interview Recursion Guide](#)

#### Core Concept Videos:

- [HackerRank Video: Recursion](#)
- [Reverse LinkedList Recursively](#)
- [Recursive Staircase Problem](#)

#### Problem Sets:

- [Problem Set: Backtracking](#)

#### Additional Videos:

- [Introducing Recursion](#)
- [Complexity of Recursion](#)
- [Limitations of Recursion](#)
- [Space Complexity Analysis of Recursion](#)
- [Modular Exponentiation](#)

### Bit Manipulation

- [InterviewCake Concept Guide](#)
- [InterviewCake Bit Manipulations](#)
- [InterviewCake Binary Numbers](#)

#### Core Concept Videos:

- [HackerRank Video: Bit Manipulation](#)
- [HackerRank: Lonely Integer Problem with Bits](#)

#### Problem Sets:

- [Problem Set: Bit Manipulation](#)

#### Additional Videos:

- Introducing Binary Number System
- Understanding Data Types in C

# 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 allotted. 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 - Compute factorial of an integer
- Challenge 2 - Compute gcd of two integers
- Challenge 3 - Integer Array Permutations
- Challenge 4 - Compute all substrings
- Challenge 5 - Single Number
- Challenge 6 - Number of 1 Bits

All the following challenges should be implemented using recursion, except where noted.

### Challenge 1 - Compute factorial of an integer

The factorial of  $n$  is the product of all integers between `1` and `n`. For example, the factorial of `5` is computed as follows: `5*4*3*2*1 == 120`.

Write a program that takes as its input a `java.math.BigInteger` and returns a `java.math.BigInteger` equal to the factorial of the input.

Hint: this can be implemented using a single static method with only one line of code.

## Challenge 2 - Compute gcd of two integers

The greatest common divisor (gcd) of two or more non-zero integers is the largest positive integer that divides all the numbers without a remainder.

Write a program that takes as its input two `int` values  $> 0$  and returns an `int` value equal to their gcd.

## Challenge 3 - Integer Array Permutations

Write a program that takes as its input an `int[]` containing 2 or more values and returning all possible permutations of the numbers.

`[1,2,3]` will have the following permutations:

```
[1,2,3]
[1,3,2]
[2,1,3]
[2,3,1]
[3,1,2]
[3,2,1]
```

Note: No two entries in the permutation sequence should be the same. For the purpose of this problem, assume that all the numbers in the collection are unique. Do not use any library functions for generating permutations.

## Challenge 4 - Compute all substrings

Consider the following interface:

```
interface SubstringProvider {
    java.util.Collection<String> getSubstrings(String s);
}
```

Write two implementations of this interface: one using an iterative algorithm and the other using a recursive algorithm.

Example: Given the input string `12345`, the output of the program would contain the following strings:

```
[1, 12, 123, 1234, 12345, 2, 23, 234, 2345, 3, 34, 345, 4, 45, 5]
```

## Challenge 5 - Single Number

Given an array of integers, every element appears twice except for one. Find that single one.

```
Input : [1 2 2 3 1]
Output : 3
```

**Note:** Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

## Challenge 6 - Number of 1 Bits

For example, the 32-bit integer 11 has binary representation:

so the **function** should return 3.

## Solutions

- [https://courses.codepath.com/courses/interview\\_prep/unit/4#!challenges](https://courses.codepath.com/courses/interview_prep/unit/4#!challenges)



# Interview Prep Course

## Interview Questions

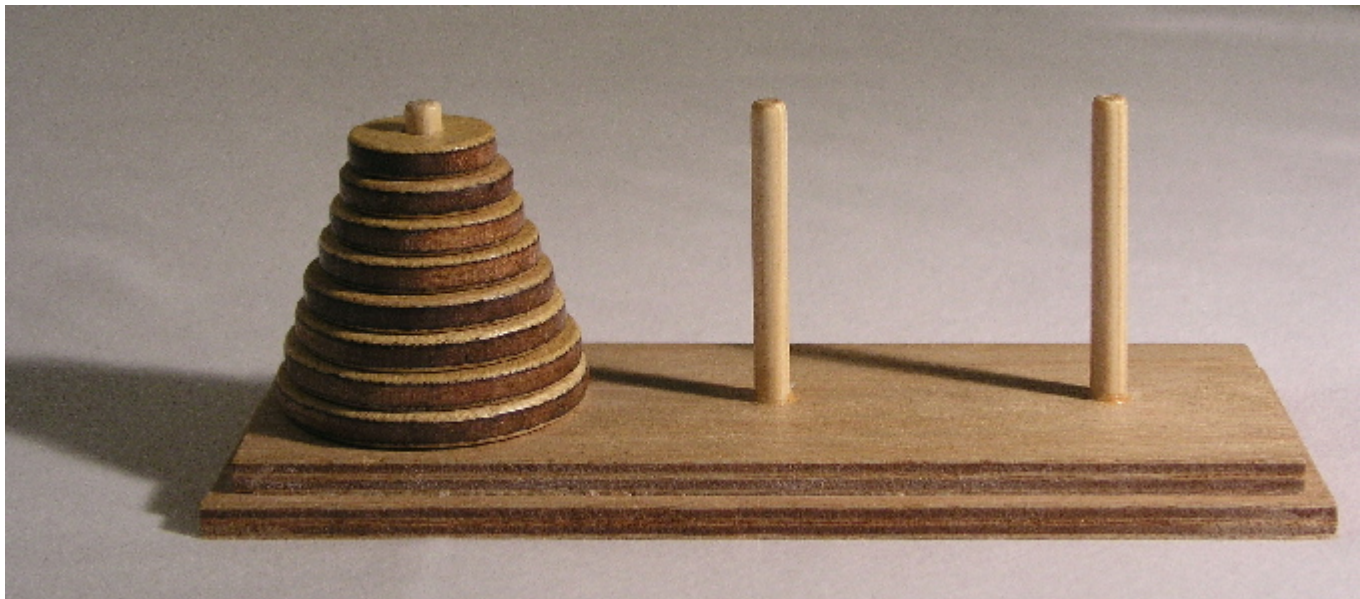
---

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 - Towers of Hanoi



In the classic Towers of Hanoi problem, there are three rods, *A*, *B*, and *C*, and there are  $n$  number of discs of different unique sizes which can slide onto any rod. The puzzle starts with all the discs in a neat stack in ascending order of size on the rod *A*, the smallest at the top, thus making a conical shape.

The objective of the puzzle is to move the entire stack to rod *C*, obeying the following simple rules:

- Only one disc can be moved at a time
- Only the uppermost disc can be moved
- No disc may be placed on top of a smaller disc

Write a program which uses recursion to compute the correct sequence of moves that solve the puzzle, using the following interface:

```
public interface TowersOfHanoiSoln {

    /**
     * All possible moves between rods A, B, and C
     */
    public static enum Move {
        AB, AC, BA, BC, CA, CB;
    }

    /**
     * Solve the sequence of correct moves for n discs from rod A to rod C.
     *
     * @param n - number of discs
     * @return the sequence moves
     */
    Move[] solve(int n);
}
```

### Bonus 1 - Big O analysis

Identify the time and space complexity of your solution. Can it be improved? If so, how? If not, why?

## Interview Question 2 - Calculating $n$ choose $k$

The *binomial coefficient*,  $\binom{n}{k}$  is often read aloud as " $n$  choose  $k$ " and is used to denote the number of different ways to choose  $k$  elements from a set of  $n$  elements without respect to their order.

There are a number of different approaches to calculating this value, including a well-known approach using recursion:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad \text{for all integers } n, k : 1 \leq k \leq n-1,$$

Stated another way, the value for ( $n$  choose  $k$ ) is equal to the sum of ( $n-1$  choose  $k-1$ ) and ( $n-1$  choose  $k$ ).

Write a program that takes as its input two *int* values  $n$  and  $k$  and implements the above algorithm using tail recursion, returning a *long* value representing the number of different ways to choose  $k$  elements from a set of  $n$  elements.

Hint: there two base cases to consider

### Bonus 1 - Memoization

The above solution is known to be inefficient due to repeated calculations. Improve the solution using memoization and explain the improvement in efficiency and the cost at which it is made.

# Interview Prep Course

## Assignment 4

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 Recursion and Bit Manipulation 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 Recursion and Bit Manipulation 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.

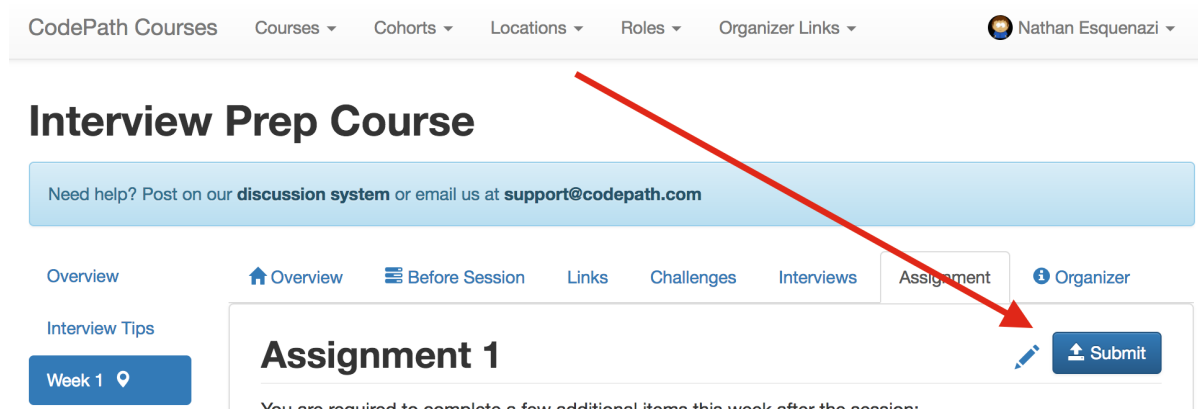
## Submitting the Assignment

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:



The screenshot shows the top navigation bar of the Interview Prep Course with links for CodePath Courses, Courses, Cohorts, Locations, Roles, and Organizer Links. Below this is a header for the "Interview Prep Course" with a help link. The main content area has a tabbed interface with "Overview", "Before Session", "Links", "Challenges", "Interviews", "Assignment", and "Organizer". The "Assignment" tab is selected, showing "Assignment 1" with a "Submit" button. A red arrow points from the "Submit" button in the screenshot to the "Submit" button in the text description.

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

Submit

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.