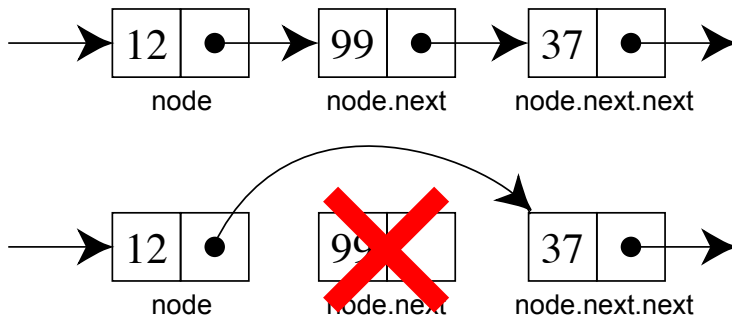


Interview Prep Course

Unit 2 - Big O, Hashtables, Linked Lists



Topics

- Big O Notation - Describing an algorithm's complexity
- Hashtables - Stores data with key value pairs
- Linked Lists - Stores data with nodes that point to other nodes

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

Tasks Before Session 2

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 Hashing and LinkedLists 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 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:

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

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.

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 - Big O, Hashtables, Linked Lists

Big-O

- InterviewCake Big-O
- Complexity Guide
- Big-O Cheat Sheet

Core Concept Videos:

- HackerRank Video: Big-O
- Derek Banas Big-O
- Log N Explained

Problem Sets:

- Problem Set: InterviewBit Complexity

Additional Videos:

- How to calculate complexity
- Asymptotic notations
- Time complexity
- General complexity rules

Hashtables

- InterviewCake Hashtable

Core Concept Videos:

- HackerRank Video: Hash Tables

Problem Sets:

- Problem Set: Hashing

Additional Videos:

- Introducing Hash Tables
- Hash Table Algorithms

Linked Lists

- InterviewCake LinkedLists
- LinkedList Succinctly Guide

Core Concept Videos:

- HackerRank Video: LinkedList
- HackerRank: Cycles in a LinkedList
- Arrays vs LinkedList

Problem Sets:

- Problem Set: LinkedList

Additional Videos:

- Intro to LinkedList
- CodeTutorial: Reverse Linked List
- Alternate Reverse a LinkedList
- Doubly Linked List
- Implementing Doubly Linked List
- Find Merge Point of Two Linked Lists
- C++ Implementation Video

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 - Implement a Linked List
- Challenge 2 - Add two numbers
- Challenge 3 - Evaluating Time Complexity
- Challenge 4 - Hash table word count
- Challenge 5 - Multimaps
- Challenge 6 - Longest non-repeating substring

Challenge 1 - Implement a Linked List

Implement a simple linked list in Java without using any collections classes. The list should be implemented using a single class such that each instance represents a single node in the list, encapsulating the node's value and a reference to the following node, as well as a convenience method to initialize a whole list from an array of values. The class should implement the following interface:

```

public interface LinkedListNode<E> {

    /* getter/setter for this node's value */
    E getValue();
    void setValue(E value);

    /* getter/setter for the subsequent node, or null if this is the last node */
    LinkedListNode<E> getNext();
    void setNext(LinkedListNode<E> next);

    /**
     * Initialize this node and all of its subsequent nodes from
     * an array of values, starting with the head value at index 0
     *
     * @param listValues - the ordered values for the whole list
     */
    void setValuesFromArray(E[] listValues);

}

```

When complete, you should be able to successfully run the following unit test using your implementation:

```

import static org.junit.Assert.*;
import org.junit.Test;

public class LinkedListNodeTest {

    @Test
    public void test() {
        LinkedListNode<Integer> head = null;

        Integer[] listValues = new Integer[] {1, 2, 3};

        head = new LinkedListNodeImpl<>(); // replace with your implementation
        head.setValuesFromArray(listValues);

        assertEquals(listValues[0], head.getValue());
        assertNotNull(head.getNext());
        assertEquals(listValues[1], head.getNext().getValue());
        assertNotNull(head.getNext().getNext());
        assertEquals(listValues[2], head.getNext().getNext().getValue());
        assertNull(head.getNext().getNext().getNext());
    }

}

```

Challenge 2 - Add two numbers

In this exercise, we'll use your `LinkedListNode` implementation to represent a non-negative integer such that each node in the list represents a single digit (in base 10) and the digits are stored in *reverse* order.

```
1      == 1
1→2    == 21
1→2→3  == 321
```

Write a program which takes as its input two such lists, a and b , and adds them arithmetically one decimal at a time. Your algorithm should traverse both lists together, adding the values for each node and carrying the 1 to the next place when the sum ≥ 10 . The result should be returned as a linked list in the same format as the input lists.

Examples:

```
1→2    +   5→3    == 6→5      // 21 + 35 == 56
1→2    +   1→2→3  == 2→4→3    // 21 + 321 == 342
1→2→3  +   7→8→9  == 8→0→3→1  // 321 + 987 == 1308
```

Challenge 3 - Evaluating Time Complexity

Identify the time and space complexity of the following using Big O notation:

- What is the time and space complexity of the `LinkedListNode.setValuesFromArray` method in Challenge #1?
- What is the time and space complexity of addition algorithm in Challenge #2?

Hint: the Big O complexity for the addition algorithm will require two variables to reflect the fact that two inputs are used

Challenge 4 - Hash table word count

Write a program which takes as its input a `String` of natural language text and outputs a `HashMap<String,Integer>` whose keys are the unique words in the input and whose values are the number of times those words occur. The algorithm should be *case-insensitive* (e.g. "Program" and "program" would count as the same word) and ignore punctuation and whitespace.

Example: Given the input `"To be or not to be, that is the question"`, the outputted `HashMap` would contain 8 entries, with two words having a count of 2 and six words having a count of 1:

```
"to"      → 2
"be"      → 2
"or"      → 1
"not"     → 1
"that"    → 1
"is"      → 1
"the"     → 1
"question" → 1
```

Challenge 5 - Multimaps

One common pattern when using hash tables requires building a `Map` whose values are `Collection` instances. In this challenge, we'll take the output of the previous challenge and invert it.

Write a program that takes as its input a `HashMap<String,Integer>` and returns a `HashMap<Integer,HashSet<String>>` containing the same data as the input map, only inverted, such that the input map's values are the output map's keys and the input map's keys are the output map's values.

Example:

Consider the example output for Challenge #4. Using that map as the input, the output for this challenge would be:

```
2 → ["to", "be"]
1 → ["or", "not", "that", "is", "the", "question"]
```

Challenge 6 - Longest non-repeating substring

Write a program which takes as its input a `String` and returns the length of the longest substring that does *not* contain any repeated characters.

Example: Given the string `"abcabcbb"`, the longest substring with no repeated characters is `"abc"`, so the program would return a value of `3`. Given the string `"aaaaaaa"`, the longest non-repeating substring is `"a"` and thus the output would be `1`.

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 - Linked List Duplicate Removal

Write a program which takes as its input an unsorted linked list of integers and deletes any duplicate values from the list *without using a temporary buffer collection* or any additional collection classed, such as a `HashSet`. Use the `LinkedListNode` implementation from this week's challenge.

Example: Given a linked list `1→2→3→3→4→4`, the program should output: `1→2→3→4`

Once you've implemented the algorithm, identify the *time and space complexity* of your solution using Big O notation.

Bonus 1

Implement an alternative solution, this time using a temporary storage buffer. This second solution should represent a significant improvement over the first in terms of time complexity, though it will come at the cost of additional space complexity. Be sure you can identify both using Big O notation.

Hint: Refer to the Big O Notation page and consult the the noted examples to determine which of them is the best fit.

Interview Question 2 - Implement a Hashtable

Implement a simple hashtable without using any special collections classes or helpers (you can use native Java arrays). Your implementation should minimally meet the following requirements:

- Support generics for both key and value types
- Implement the standard `get`, `put`, `remove`, `size`, `clear`, and `isEmpty` operations as defined in `java.util.Map`
- Support an initial default capacity of 16 entries

- Support dynamic allocation of additional capacity as needed

Bonus 1

Implement the entire `java.util.Map` interface by adding support for the remaining operations:

- Implement `containsKey` and `containsValue`
- Implement `keySet`, `keySet`, and `values`
- Implement `putAll`

Hint: One standard approach to this problem would involve using two-dimensional arrays.

Interview Prep Course

Assignment 2

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 Hashing and LinkedLists 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 Hashing and LinkedLists 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, Organizer Links, and a user profile for Nathan Esquenazi. Below this is the course title "Interview Prep Course" and a help banner. The main content area has a tabbed interface with "Overview", "Before Session", "Links", "Challenges", "Interviews", "Assignment", and "Organizer". The "Assignment" tab is selected. Below the tabs, there's a section titled "Assignment 1" with a description: "You are required to complete a few additional items this week after the session:". To the right of this section is a "Submit" button with an upload icon. A red arrow points from the "Submit" button in the screenshot to the "Submit" button in the diagram below.

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.