Cracking the Coding Interview

Alex Ni - July 22nd, 2022

Alex Ni

- Senior consultant at Redapt, Advanced Analytics
- 4+ years of experience in ETL, BI, data warehouse
- Familiar with data science and full cycle machine learning pipeline
- Certified in Azure data and AI scientist and GCP data engineer and cloud architect
- Connect on LinkedIn:

  https://www.linkedin.com/in/haowen-ni-860612117

Agenda

- What to expect in a coding interview
- How to prepare for coding interview
- Demo: Two Sum

CRACKING the CODING INTERVIEW
189 PROGRAMMING QUESTIONS & SOLUTIONS
GAYLE LAAKMANN MCDOWELL
6TH EDITION

[CRACKING the CODING INTERVIEW - Home (crackingthecodinginterview.com)]

What is the coding interview?

- A 45 minutes, problem-oriented session used to assess potentially employees on:
    - Analytical skills
    - Coding Skills
    - Technical/CS knowledge and fundamentals
    - Communication skills
- Technical phone interviews
    - Technical challenges using web-based code editor like Coderpad and HackerRank
- Onsite interviews
    - 3 to 5 rounds technical interviews
    - Coding on a whiteboard
- Online assessment/Take home
    - For example, Amazon OA

Research on what questions each company will ask me

- **IMPORTANT**: Figure out in advance what common question are asked for the company you're interviewing with
- Generally, you can find this information in a few places:
  - **Job description**: to gauge the focus of the questions
  - **Coding interview prep websites**: for example, Leetcode has collections of top interview questions for each big tech company
  - **Current/former employees**: online forums, blogs, places where they discuss about their interview process
  - **Recruiter**: ask them directly for topics that may appear
- Eliminate the guesswork from coding interviews:
  - https://www.codinginterview.com

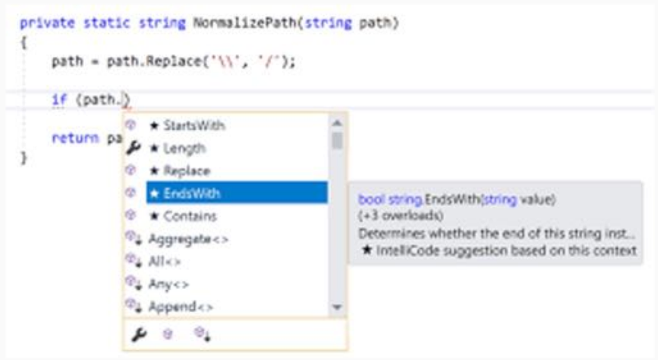
How to prepare for coding interview?

Tip1: **DON’T** jump into coding questions directly

- **Foundations in coding**
  - The knowledge you will learn from CS101: variables, data type, functions, parameters, for/while loops, if/else, recursion, and etc.,
- **Data structures and algorithms**
  - Arrays, Linked Lists, Hash Tables, Stacks and Queues, Trees, Graphs, ...
  - Binary search, Sorting, Breadth First Search, Depth First Search, …
- **Complexity Analysis, Big O notation**
  - **Time Complexity:** how fast an algorithm runs, expressed using
  - **Space Complexity:** how much memory an algorithm takes up

Tip 2: Pick the programming language most **comfortable** with you

- You need to be extremely familiar with the syntax so that you can better focus on the logic of solution
  - No intellisense such as code completion and syntax checking
  - Coding style



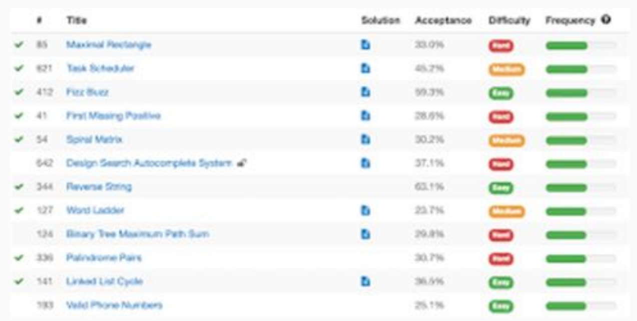
```
private static string NormalizePath(string path)
{
    path = path.Replace('\\', '/');

    if (path.)

    return pa
}
```

bool string.EndsWith(string value)
(+3 overloads)
Determines whether the end of this string inst...
★ IntelliCode suggestion based on this context

Tip 3: **MOST IMPORTANT: Practice, practice, and practice**

- Make a continuous plan for your preparation
  - Study for the basic knowledge
  - Begin with easier questions to brush up what you just learned about data structures and algorithms knowledge
  - Start to timing yourself, think out loud, consider time and memory complexity
  - Practice more complex coding problems
- Practice on LeetCode
  - If you practice enough LeetCode questions, there is a good chance that you will either see or complete one of your actual interview questions (or some variant of it)

LeetCode



	#	Title	Solution	Acceptance	Difficulty	Frequency
✓	85	Maximal Rectangle	🗎	33.0%	Hard	🟩
✓	621	Task Scheduler	🗎	45.2%	Medium	🟩
✓	412	Fizz Buzz	🗎	59.3%	Easy	🟩
✓	41	First Missing Positive	🗎	28.6%	Hard	🟩
✓	54	Spiral Matrix	🗎	30.2%	Medium	🟩
	642	Design Search Autocomplete System 🔒	🗎	37.1%	Hard	🟩
✓	344	Reverse String		63.1%	Easy	🟩
✓	127	Word Ladder	🗎	23.7%	Medium	🟩
	124	Binary Tree Maximum Path Sum	🗎	29.8%	Hard	🟩
✓	336	Palindrome Pairs		30.7%	Hard	🟩
✓	141	Linked List Cycle	🗎	36.5%	Easy	🟩
	193	Valid Phone Numbers		25.1%	Easy	🟩

Demo: Two Sum

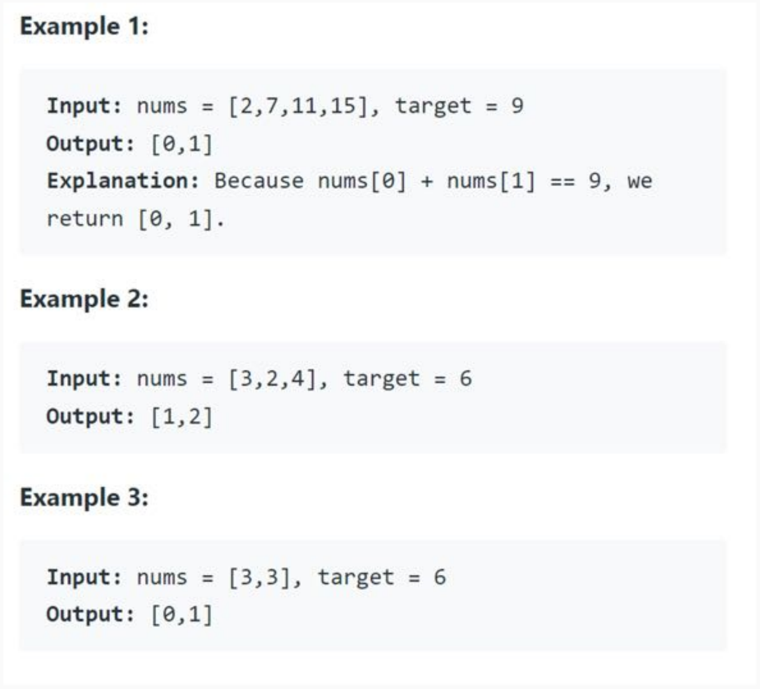
Two Sum - Description

LeetCode #1

Given an array of integers numbers and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.



Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we
return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

Two Sum - Hints

Hint 1

A really brute force way would be to search for all possible pairs of numbers but that would be too slow. Again, it's best to try out brute force solutions for just for completeness. It is from these brute force solutions that you can come up with optimizations.

Approach 1: Brute Force

Algorithm

The brute force approach is simple. Loop through each element x and find if there is another value that equals to target - x.

Time complexity:

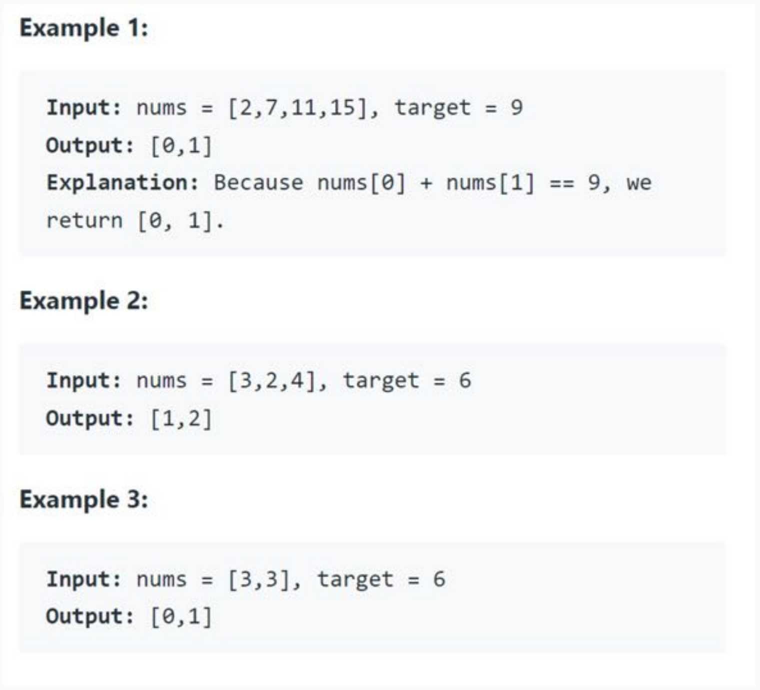
O(n^2). For each element, we try to find its complement by looping through the rest of the array which takes O(n) time. Therefore, the time complexity is O(n^2).

Space complexity:

O(1). The space required does not depend on the size of the input array, so only constant space is used.

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        for i in range(len(nums)):
            for j in range(i + 1, len(nums)):
                if nums[j] == target - nums[i]:
                    return [i, j]
```

Two Sum - Hints



Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we
return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

Hint 2

So, if we fix one of the numbers, say x

We have to scan the entire array to find the next number y, which is value - x where value is the input parameter.

Can we change our method somehow so that this search becomes faster?

Two Sum - Hints

Hint 3

The second train of thought is, without changing the array, can we use additional space somehow? Like maybe a hash map to speed up the search?

Approach 2: Two-pass Hash Table

Algorithm

A simple implementation uses two iterations. In the first iteration, we add each element's value as a key and its index as a value to the hash table. Then, in the second iteration, we check if each element's complement (target - nums[i]) exists in the hash table. If it does exist, we return current element's index and its complement's index. Beware that the complement must not be nums[i] itself!

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        hashmap = {}
        for i in range(len(nums)):
            hashmap[nums[i]] = i
        for i in range(len(nums)):
            complement = target - nums[i]
            if complement in hashmap and hashmap[complement] != i:
                return [i, hashmap[complement]]
```

Approach 2: Two-pass Hash Table

Time complexity:

O(n). We traverse the list containing nn elements exactly twice. Since the hash table reduces the lookup time to O(1), the overall time complexity is O(n).

Space complexity:

O(n). The extra space required depends on the number of items stored in the hash table, which stores exactly n elements.

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        hashmap = {}
        for i in range(len(nums)):
            hashmap[nums[i]] = i
        for i in range(len(nums)):
            complement = target - nums[i]
            if complement in hashmap and hashmap[complement] != i:
                return [i, hashmap[complement]]
```

Approach 3: One-pass Hash Table

Algorithm

It turns out we can do it in one-pass. While we are iterating and inserting elements into the hash table, we also look back to check if current element's complement already exists in the hash table. If it exists, we have found a solution and return the indices immediately.

Time complexity:

O(n). We traverse the list containing nn elements only once. Each lookup in the table costs only O(1) time.

Space complexity:

O(n). The extra space required depends on the number of items stored in the hash table, which stores at most n elements.

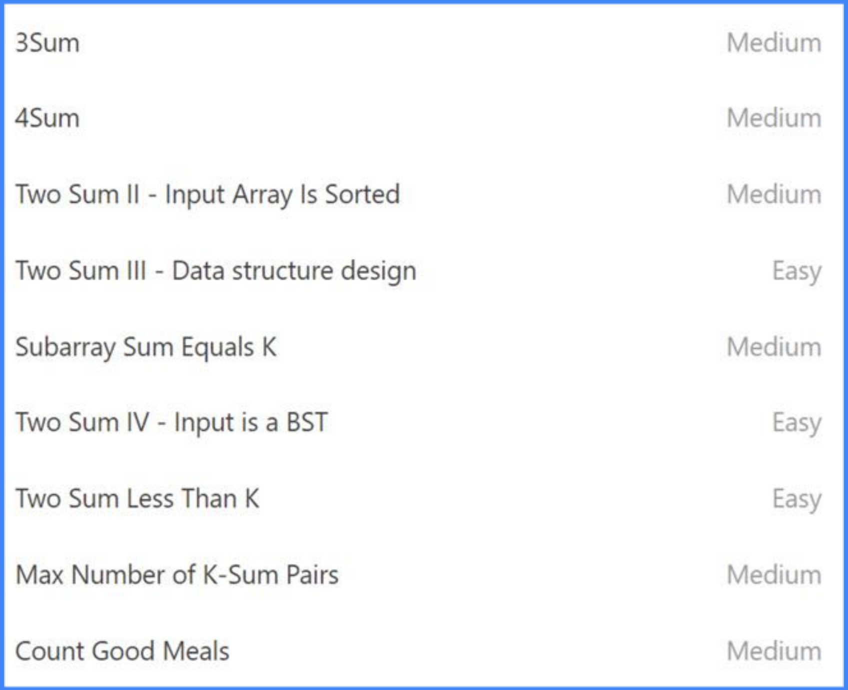
```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        hashmap = {}
        for i in range(len(nums)):
            complement = target - nums[i]
            if complement in hashmap:
                return [i, hashmap[complement]]
            hashmap[nums[i]] = i
```

Two Sum - Similar Questions

LeetCode #15 - 3 Sum

Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.

Notice that the solution set must not contain duplicate triplets.



3Sum	Medium
4Sum	Medium
Two Sum II - Input Array Is Sorted	Medium
Two Sum III - Data structure design	Easy
Subarray Sum Equals K	Medium
Two Sum IV - Input is a BST	Easy
Two Sum Less Than K	Easy
Max Number of K-Sum Pairs	Medium
Count Good Meals	Medium

Summary

For preparing coding interviews, remember:

- Start with the foundational knowledges, especially data structures and algorithms
- Pick the programming language most comfortable with you
- Start with the easy questions, build a long term motivation and keep practicing

Thank you!

Reference

- Cracking the coding interview
  - https://www.crackingthecodinginterview.com/
- LeetCode - The World's Leading Online Programming Learning Platform
  - https://leetcode.com/
- Coding Interview Preparation | Codinginterview
  - https://www.codinginterview.com/