BloomTech’s Downfall: A Long Time Coming

Your source for the latest news and trends in online education.

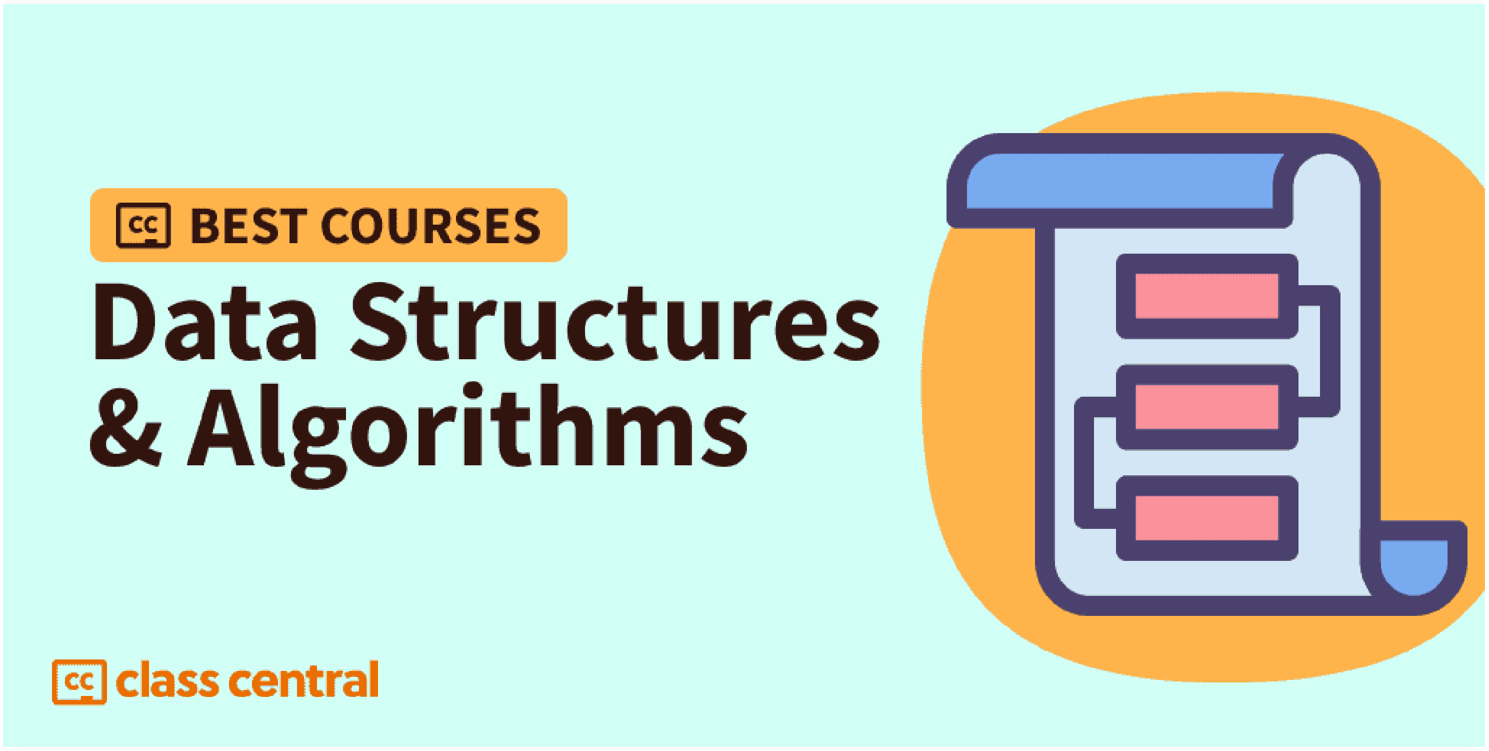
BEST COURSES GUIDES

10 Best Data Structures & Algorithms Courses to Take in 2024

Ten carefully-chosen courses to help you become a better coder and prepare for technical interviews.

Elham Nazif
Apr 18th, 2022

6



Data structures and algorithms are essential to if you want to write efficient code, which is why they're a fundamental part of technical interviews.

In this guide, I've leveraged Class Central's catalog of 100K courses to find the best data structure and algorithm courses available online.

You can read all about my ranking methodology below. But if you're in a hurry, here are my top picks — click on a course to skip to the details:

Course	**Workload**	**In Brief**
1. Algorithms, Part I (Princeton)	60 hours	Best overall course in Java that covers all you need to know
2. Algorithms: Design and Analysis (Stanford)	24 hours	Best language-agnostic course that isn't afraid of math
3. Algorithms and Data Structures Tutorial (Treehouse)	6 hours	Best Python course for beginners who are shaky with math
4. Introduction to Algorithms (MIT)	36 hours	Immensely rigorous but rewarding Python course for those who want a challenge
5. Data Structures & Algorithms (Georgia Tech)	50 hours	Rigorous Java course with focus on data structures
6. Algorithmic Toolbox (UC San Diego)	40 hours	Language-agnostic and highly practical course focused on algorithms
7. JavaScript Algorithms and Data Structures (freeCodeCamp)	100–200 hours	Great for JS beginners — with **free certificate**
8. Data Structures and Algorithms in Python (Jovian)	13 hours	For those comfortable in Python programming

Course	Workload	In Brief
9. Master the Coding Interview: Data Structures + Algorithms (Udemy)	20 hours	Great language-agnostic course to prepare for technical interviews
10. Intro to Data Structures and Algorithms (Google)	4 weeks	Prepares you for technical interviews with Python

What are Data Structures and Algorithms?

Data structures and algorithms are everywhere, and there is a good reason for that. Data structures help us hold data and provide affordances for efficiently manipulating it, while algorithms represent tried-and-tested computational recipes to accomplish particular goals. Let me explain how it all works.

Human thinking may seem flexible and seemingly unconstrained, but in truth, we have a flurry of biological limitations. For instance, we typically can only hold in our short-term memory about 7 items — the famous magical number 7.

Well, computers have similar limitations. They have limited resources that constrain what can be accomplished with software in a number of ways: computers can only store so much data in memory, and computers can only perform so many operations per second — most notably, CPUs.

That said, there are many ways of taking advantage of computer resources, some more efficient than others. Data structures in particular, as their name suggests, allow to organize information in such a way that it can efficiently be leveraged in certain contexts. There are many data structures, each with their strengths and weaknesses. A good developer should know the fundamental data structures and when to use each. They're essential components of many applications, and they're ubiquitous in high-performance computing.

An algorithm is a series of step-by-step instructions that aims to achieve a particular goal. They're general methods for tackling similar problems — for example, finding a name in a sorted list of names, or finding the shortest path between two nodes in a graph.

Data structure and algorithms often work hand-in-hand. A particular algorithm typically leverages a particular data structure. Well thought out combinations of algorithms and data structures can help minimize the running time and memory needed to perform certain operations, thus saving a lot of time and money in the process.

Having a wide arsenal of data structures and algorithms at your disposal grants you the freedom to choose the most appropriate solution to code whatever you want.

Another major reason why you should learn data structures and algorithms is that many giant tech companies like Microsoft and Google include them in their interviews. During technical interviews, you'll be asked to solve computing problems that don't simply involve picking an appropriate data structure or algorithm but also implementing them and showing the interviewer that you indeed understand their strengths and weaknesses.

Course Ranking Methodology

I built this ranking following the now tried-and-tested methodology I used in previous rankings (you can find them all here). It involves a three-step process:

First, let me introduce myself. I'm part of the Class Central team, and I (@elham) built this ranking in collaboration with my friend and colleague @manoel.

We began with a purely data-driven process by leveraging Class Central's database of 100K courses to make a preliminary selection of DSA courses. We took a look at things like ratings, reviews, and course bookmarks to bring you some of the most popular DSA courses.

But we didn't stop there. Ratings and reviews rarely tell the whole story. So the next step was to bring our personal knowledge of online education into the mix.

Second, we used our experience as online learners to evaluate each preliminary pick.

Both of us come from computer science backgrounds and are prolific online learners, having completed about 45 MOOCs between us. Additionally, Manoel has an online bachelor's in computer science, while I am currently completing my foundation in computer science. So we know our way around algorithms and data structures!

By carefully analyzing each course and bouncing ideas off each other, we've made iterative improvements to the rankings until we were both satisfied.

Third, during our research, we stumbled across courses that we felt were well-made but weren't well-known. Had we adopted a purely data-centric approach, we would be forced to leave those courses out of the ranking just because they had fewer enrollments.

To avoid this, we have decided instead to take a more holistic approach. We spiced up the ranking by including a wide range of courses on data structure and algorithms that will hopefully cater to different reader's preferences.

After going through this process — combining Class Central data, our experience as lifelong learners, and a lot of editing — we arrived at our final ranking. So far, we've spent more than 10 hours building this ranking, and we intend to continue updating it in the future.

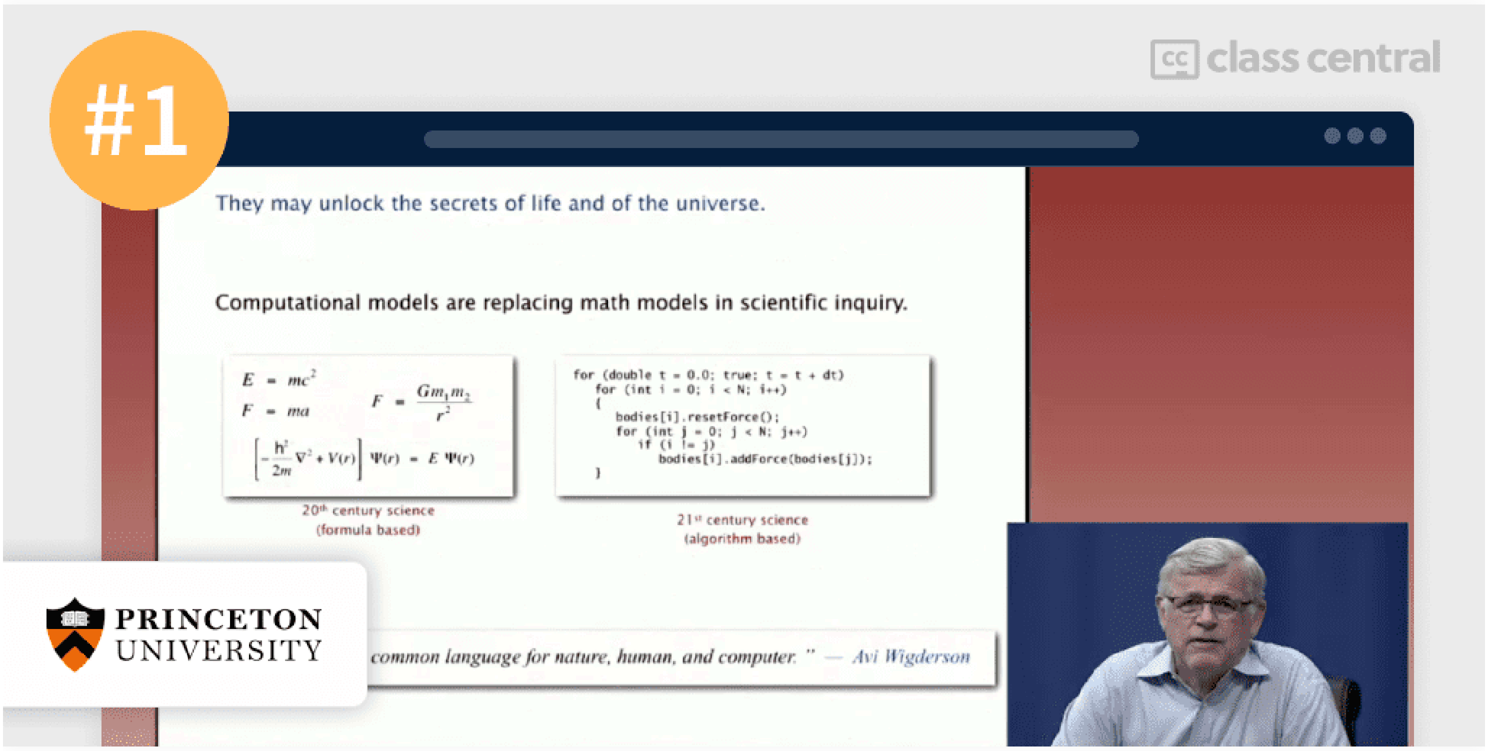
Course Ranking Statistics

Here are some aggregate stats about the ranking:

- In this ranking, the largest course in terms of enrolments has around 931K students.
- Combined, the courses have a total of 1.6M enrollments.
- 9 of the courses are free or free-to-audit, whereas only 1 course is paid.
- About 209K+ students are following Algorithms and Data Structures Courses on Class Central.

Now that the data nerds have been satisfied, let's get to the top picks!

1. Algorithms, Part I (Princeton University)



They may unlock the secrets of life and of the universe.

Computational models are replacing math models in scientific inquiry.

$$E = mc^2 \qquad F = ma \qquad F = \frac{Gm_1m_2}{r^2}$$

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V(r)\right]\Psi(r) = E\,\Psi(r)$$

20th century science (formula based)

```
for (double t = 0.0; true; t = t + dt)
   for (int i = 0; i < N; i++)
   {
      bodies[i].resetForce();
      for (int j = 0; j < N; j++)
         if (i != j)
            bodies[i].addForce(bodies[j]);
   }
```

21st century science (algorithm based)

common language for nature, human, and computer. " — Avi Wigderson

Robert Sedgewick, course co-instructor

My #1 pick for the best DSA course has to be *Algorithms, Part I* by Princeton University.

This Coursera course touches on elementary data structures, sorting, and searching algorithms. It covers the essential information that every serious programmer needs to know about algorithms and data structures, with emphasis on applications and scientific performance analysis of Java implementations.

To take this course, you'll need some familiarity with Java, including loops, arrays, functions, recursion, and objects, as well as high-school algebra.

What You'll Learn

The course covers three main topics: data types, sorting, and searching.

A data type is an attribute of data which shows how a programmer intends to use the data. The course introduces a variety of data types. You'll first learn about the union-find data type by considering the dynamic connectivity problem — given a grid, is there a connected path between points A and B?

Next, you'll study the fundamental data types for storing collections of objects: the stack and the queue. You'll learn how to implement each using either a singly-linked list or a resizing array, and then consider various applications of stacks and queues ranging from parsing arithmetic expressions to simulating queueing systems.

You'll also learn about the priority queue data type, along with its implementation using the binary heap data structure. You'll discuss the applications of priority queues by simulating the motion of n particles.

An algorithm that orders the elements of a list is called a sorting algorithm. Starting with the elementary sorting methods (selection sort and insertion sort), you'll consider two algorithms for uniformly shuffling an array. Moving on to more advanced algorithms, you'll study mergesort and quicksort. You'll analyze their performance and then compare the best situations to use them. You'll also learn about heapsort, which uses the binary heap data structure, as well as radix sorts.

While sorting algorithms order elements, searching algorithms work to retrieve information stored within data structures. To guarantee a logarithmic performance for search and insert, you'll develop a symbol table which associates values with keys (think of associative arrays, or dictionaries).

Lastly, you'll learn about hash tables and hash functions, which allow you to map keys and values for efficient retrieval. You'll describe what a good hash function should have, and then learn to implement them in Java. Then, you'll consider two strategies for implementing hash tables to yield constant-time performance for search and insert.

How You'll Learn

The course is 6 weeks long, with a study commitment of roughly 6–10 hours per week. There are two lectures per week, each broken up into about 4–6 segments and separated by interactive quiz questions to help you process and understand the material.

Regarding assessments, there are 5 lengthy coding assignments about data structures and algorithms to complete. Additionally, the course also provides a few algorithmic job interview questions based on the material for the week which are inspired by questions asked at leading technology companies.

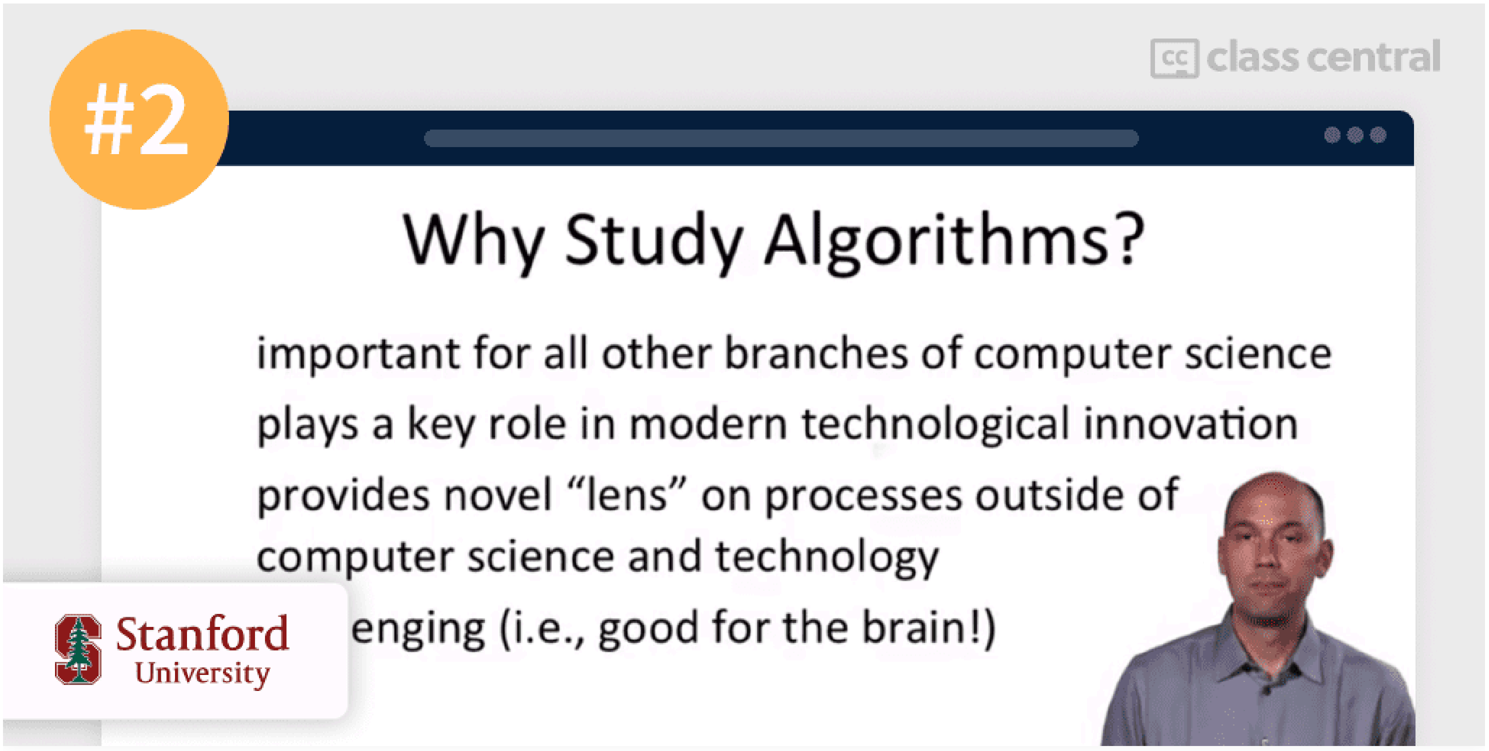
Institution	Princeton University
Provider	Coursera
Instructors	Robert Sedgewick and Kevin Wayne
Level	Intermediate
Workload	60 hours total
Enrollments	929K
Rating	4.9 / 5.0 (9.4K)

Fun Facts

- The course has 19.4K bookmarks on Class Central.
- The next part of the course is Algorithms, Part II, focusing on graph-processing and string-processing algorithms, as well as reductions and intractability.
- This course can also be accompanied by the free online book Algorithms, 4th Edition, of which the instructors are co-authors.

If you're interested in this course, you can find more information about the course and how to enroll here.

2. Algorithms: Design and Analysis, Part 1 (Stanford University)



Why Study Algorithms?

important for all other branches of computer science
plays a key role in modern technological innovation
provides novel "lens" on processes outside of computer science and technology
enging (i.e., good for the brain!)

Tim Roughgarden, course instructor

My 2nd pick for the best DSA course is *Algorithms: Design and Analysis, Part 1*, offered by Stanford University on edX.

In *Algorithms: Design and Analysis, Part 1* you will learn several fundamental principles of algorithm design and the data structures they rely on. This course is demanding but rewarding. My colleague @dhawal wrote about it here.

It emphasizes the big picture and conceptual understanding over low-level implementation and mathematical details. By the end of the course, you'll be well-positioned to ace your technical interviews and speak fluently about algorithms with other programmers and computer scientists.

The prerequisites of this course include basic programming experience in any programming language and some knowledge of mathematical proofs (proof by induction, proof by contradiction, etc).

What You'll Learn

Specific topics the course covers include: 'Big-O' notation, sorting and searching, divide and conquer, randomized algorithms, data structures, and graph primitives.

The course begins by discussing algorithms in general and why they're so important by using the problem of multiplying two integers to show how algorithmic ingenuity can greatly improve upon a naïve implementation. Then, the course discusses Merge Sort as a warm up for the more intricate algorithms that'll follow.

Big O notation belongs in the vocabulary of every serious programmer and computer scientist. The goal of this section is to focus on how the running time of any algorithm scales as the input size grows larger, and then compare the efficiency of different algorithms.

Then, you'll learn the divide-and-conquer design paradigm and how it can speed up various applications like fast sorting, searching, and matrix multiplication. Afterwards, you'll cover a "black-box" method for solving recurrences. You'll then be able to determine the running time of most of the divide-and-conquer algorithms you'll ever see!

After that, you'll study the problem of computing the ith smallest element of an input array. By using the Quick Sort algorithm instead of a naïve approach, you'll do this in linear time instead of O(n log n) time — a vast improvement!

The course then shifts gears to graph theory. First, you'll review graphs and the most standard ways of representing them (using adjacency lists). You'll also learn a simple but useful trick for transforming an algorithm that almost always fails into one that almost always succeeds. Then, you'll cover a selection of fundamental primitives for reasoning about graphs and graph search, which are all blazingly fast! All this buildup has been in anticipation of one of the most-famous algorithms ever — Dijkstra's shortest-path.

The final few sections of the course consists of lessons on data structures — heaps, binary search trees, and hash tables. The aim is to teach you the operations that these data structures support (along with their running times), as well as to develop your intuition about which data structures are useful for which sorts of problems.

You'll end with a bang by learning how heaps and binary search trees are really useful (they are used in Dijkstra's algorithm) and also how constructing effective hash functions enables hash tables to search in linear time.

One thing to note is that the course contains a lot of optional lectures that dive deeper into the theory behind the algorithms you've learnt, if you'd like to satisfy your intellectual hunger.

How You'll Learn

This course is 6 weeks long with an estimated workload of 2–4 hours per week, where you'll learn through video lectures and slides.

Regarding assessments, you'll practice and master the fundamentals of algorithms through several types of assignments. There are 6 multiple-choice quizzes to test your understanding of the most important concepts along with 6 programming assignments, where you'll implement one of the algorithms covered in the lectures in any programming language you want. You can only complete these assessments if you're paying for the certificate.

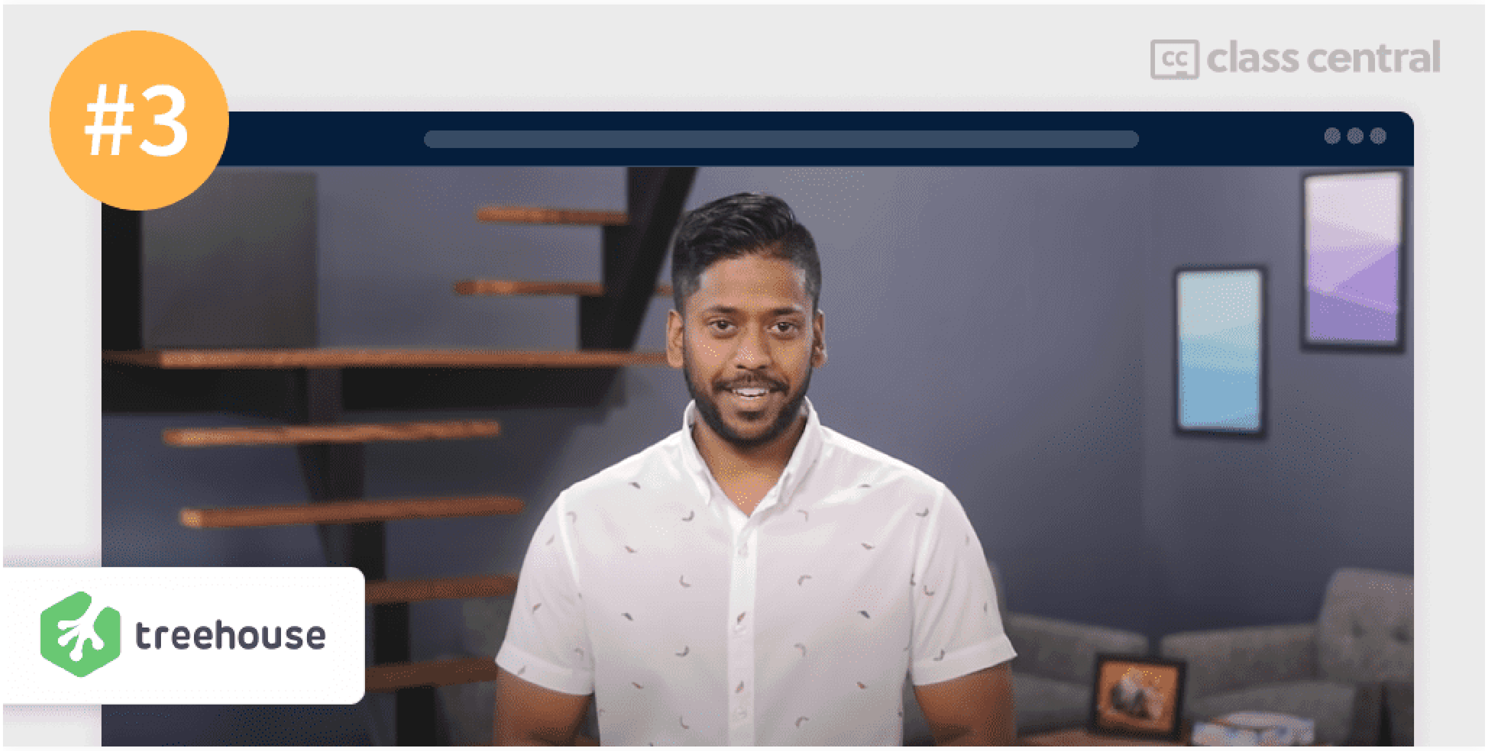
Institution	Stanford University
Provider	edX
Instructor	Tim Roughgarden
Level	Intermediate
Workload	24 hours total
Enrollments	45K
Certificate	Paid

Fun Facts

- The course has 913 bookmarks on Class Central.
- The next course you should take after this is Algorithms: Design and Analysis, Part 2.
- Tim Roughgarden is the Professor of Computer Science and member of the Data Science Institute at Columbia University.
- He received the Gödel Prize in 2012 for his work on routing traffic in large-scale communication networks to optimize performance of a congested network.
- My colleague @dhawal also wrote about the course here!

If you're interested in this course, you can find more information about the course and how to enroll here.

3. Algorithms and Data Structures Tutorial (Treehouse)



Pasan Premaratne, course co-instructor

My third pick for the best algorithms and data structures course is the *Algorithms and Data Structures Tutorial* offered by Treehouse on freeCodeCamp's YouTube channel.

This course is excellent for beginners. It teaches you all about algorithms and data structures, giving you the intuition behind each algorithm and data structure while avoiding flooding you with mathematical details.

By the end of the course, you will understand what algorithms and data structures are, how they are implemented and evaluated, and how they are used to solve problems.

What You'll Learn

There are three main parts to this course: algorithms, data structures, and a deep dive into sorting and searching algorithms.

The course begins by answering the question: What is an algorithm? An algorithm is a set of steps a program takes to finish a task. You'll see how this seemingly simple definition actually forms the foundation of executing many real-world applications. Your first introduction to algorithms comes in the game: Guess the number! Then, you'll move onto more search algorithms, like linear search and binary search.

Given that there may be multiple algorithms to finish a task, how do we find the one which saves us the most time in the worst-case or best-case scenario? This question of time complexity is solved by the Big O notation. You'll learn how we use the Big O notation to measure and determine how long an algorithm is estimated to finish given some input, and describe the time taken as linear, quadratic, quasilinear, or exponential time.

Then, you'll get to coding linear and binary search in Python. You'll learn more about the fundamental concept of recursion when implementing recursive binary search. You'll also need to keep in mind space complexity — how much storage is needed for an algorithm to function.

After that, you'll be introduced to your first data structure: arrays. You'll learn what arrays are, as well as how to access, search, insert and delete values from them. You'll also learn another important data structure called linked lists. You'll see how linked lists are more flexible than arrays, and compare the differences between them. You'll study the merge sort algorithm, how to sort a linked list with them, and how to evaluate its efficiency.

The course then looks at sorting and searching algorithms, and how to implement them in Python. You'll learn and code Bogosort, Quicksort, Merge Sort, and also gain a different perspective on Linear Search and Binary Search. You'll learn how to measure the Big-O run time of each of these algorithms with Python. Lastly, you'll see how these algorithms can be used for real-world applications.

How You'll Learn

This course will take you approximately 6 hours to complete, and consists of several video lectures filled with visualizations. You are expected to follow along with the course instructors while they are coding to further cement your understanding.

Organization	Treehouse
Provider	freeCodeCamp
Instructors	Pasan Premaratne and Jay McGavren

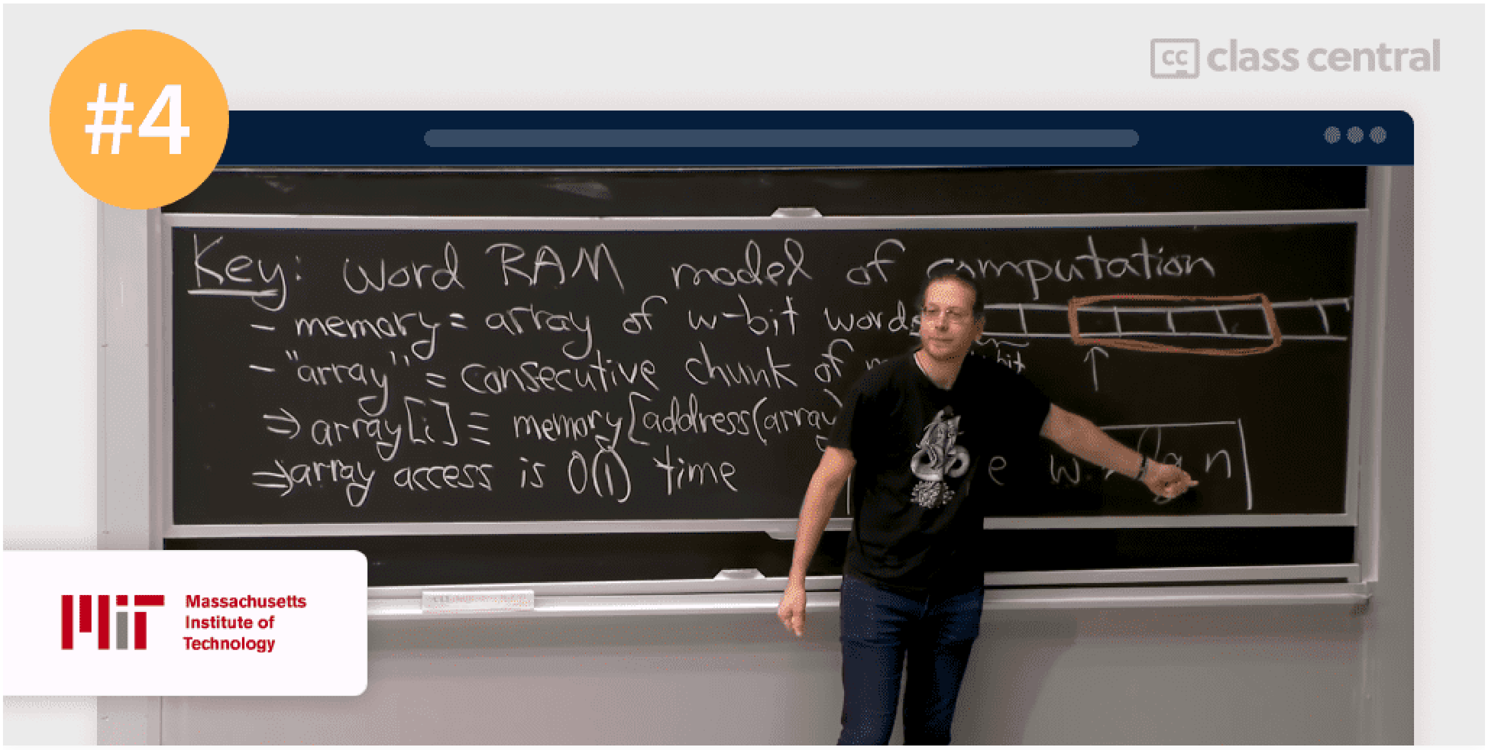
Level	Beginner
Workload	6 hours long
Views	1.7M
Likes	57K
Certificate	None

Fun Facts

- Treehouse offers many other programming courses on their website.
- Pasan is a iOS, Swift and Computer Science teacher at Treehouse, while Jay is the author of Head First Ruby and Head first Go, both published by O'Reilly Media.

If you're interested in this course, you can find more information about the course and how to enroll here.

4. 6.006 Introduction to Algorithms (Massachusetts Institute of Technology)


#4
class central
Key: Word RAM model of computation
– memory = array of w-bit words
– "array" = consecutive chunk of
⇒ array[i] = memory[address(array
⇒ array access is O(1) time
Massachusetts Institute of Technology

Erik Demaine, course instructor

6.006 Introduction to Algorithms is an introduction to mathematical modeling of computational problems. It is demanding (as expected from an MIT course) but also rewarding if you're up for the challenge.

It covers elementary data structures (dynamic arrays, heaps, balanced binary search trees, hash tables) and algorithmic approaches to solve classical problems (sorting, graph searching, dynamic programming). The course emphasizes the relationship between algorithms and programming and introduces basic performance measures and analysis techniques for these problems.

To take this course, you'll need basic experience programming in Python 3 and basic knowledge of discrete mathematics. You can try completing Problem Set 0 to see if this course is right for you.

What You'll Learn

The course starts off with the goal of teaching you how to solve computation problems and communicate that your solutions are correct and efficient. You'll learn what algorithms are and how to measure how fast it takes for them to produce a correct output using the Big O notation. You'll also learn that data structures are a way for programs to store data, with algorithms that support operations on the data.

Then, you'll learn about sets and sorting. Sorting a set of elements is important because one of the fastest searching algorithms, binary search, presupposes that the set of elements has already been sorted. Hence, you'll learn about Permutation Sort, Selection Sort, Insertion Sort, and Merge Sort. But if you want faster search and dynamic operations, you'll need to learn about hashing, chaining, and hash functions, to come up with a hash table that accesses data in linear time. This will allow you to also achieve a faster sort by using algorithms like Tuple Sort, Counting Sort, and Radix Sort!

The course then moves on to binary trees. You'll study their terminology, variations, as well as how to navigate and operate them. A binary tree is balanced if it maintains O(log n) height under dynamic operations, and AVL trees are an example of that. This will lead you to learning about binary heaps and priority queues.

Afterwards, you'll be given a lesson on graph theory and its terms. Graphs are everywhere from road networks to chess. The most common graph representation is by adjacency lists. With this, you'll learn and understand the difference between breadth-first search and depth-first search, which are the elementary search algorithms that make up the more complex ones.

The next few lessons focus on finding the shortest-path weights in weighted graphs. You'll cover the Bellman-Ford algorithm, Dijkstra's algorithm, APSP algorithm and Johnson algorithm. You'll understand the underlying concepts and compare the running time of each of these algorithms.

The final classes introduce you to dynamic programming and its features. Dynamic programming is a weird term coined by Richard Bellman, who wanted government funding but needed a cool name to disguise doing mathematics!

You'll learn and design your own recursive algorithms. For example, you'll code the Fibonacci sequence, alternating coin names, arithmetic parenthesization, and even piano fingering. Lastly, you'll relate what you've learned about complexity with polynomial and pseudo polynomial time which are characteristic of some dynamically-programmed algorithms.

How You'll Learn

The course has 13 weeks of lectures, and the course videos add up to about 36 hours. The video lectures are uploaded to YouTube. The course also includes supplemental resources like course notes, course assignments and their solutions.

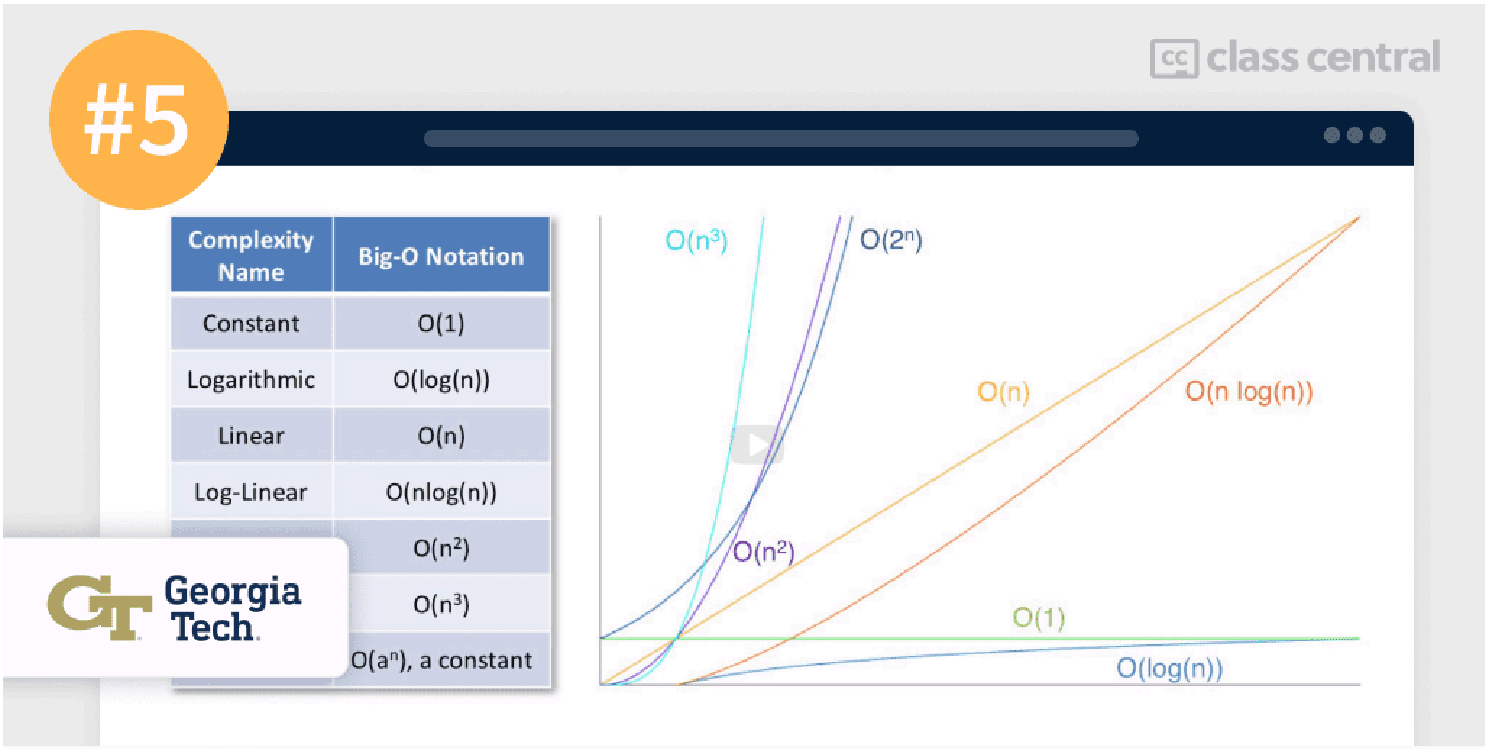
Institution	Massachusetts Institute of Technology
Provider	MITOpenCourseWare
Instructors	Erik Demaine
Level	Intermediate
Workload	36 hours total
Views	405K
Rating	9K
Certificate	None

Fun Facts

- Erik Demaine is a professor of Computer Science at the Massachusetts Institute of Technology and a former child prodigy.
- He completed his bachelor's degree at the age of 14 and completed his PhD by the time he was 20 years old!
- He is reportedly the youngest professor in the history of MIT.

If you're interested in this course, you can find more information about the course and how to enroll here.

5. Data Structures & Algorithms I: ArrayLists, LinkedLists, Stacks and Queues (Georgia Institute of Technology)



Complexity Name	Big-O Notation
Constant	O(1)
Logarithmic	O(log(n))
Linear	O(n)
Log-Linear	O(nlog(n))
	$O(n^2)$
	$O(n^3)$
	$O(a^n)$, a constant

Big O notation and growth rate

Data Structures & Algorithms I aims to empower computer scientists in training like you with the actual building blocks to create data structures and algorithms that make programs come alive.

Available on edX, this course provides an overview of basic linear data structures and the algorithms that operate on those structures. By the end of the course, you'll understand the fundamental principles of linear data structures and you'll be able to differentiate between those linear structures, implement them efficiently, and analyze their performance.

The prerequisites for the course are basic knowledge of the Java programming language and object oriented principles.

What You'll Learn

You will examine ArrayLists, LinkedLists, Stacks, Queues, and Deques, and analyze their operations and time complexity.

In the first module, you'll review important Java principles involved in object-oriented design, from constructors to references. You'll delve into the challenging concept of Big-O and how to measure time complexity at a high level. Lastly, you'll discern when to use Iterator and Iterable, and understand how to use comparable and comparator in practical Java code.

The second module introduces you to Arrays & ArrayLists. After a review of the concept of recursion and exploring recursion's principal function within data structures, you'll implement recursive methods on them. Additionally, you'll be able to describe the behavior of the data structure without concerning yourself with the low-level implementation.

Linked Lists is what the third module will cover. You'll study the basics of linked data structures by learning about the Singly-Linked List, an implementation of the List abstract data type, and learn about variations of the Linked List concept: Doubly-Linked Lists and Circularly-Linked Lists. You'll then practice the ideas of iteration and recursion on Linked Lists both conceptually and through code.

The fourth and final module will teach you about Stacks and Queues. You'll start by understanding the concepts of two fundamental abstract data types, Stacks and Queues, which you'll then implement using Arrays and LinkedLists with minimal overhead. This brings us to Priority Queues and Deques, variations of Stacks and Queues. You'll apply their conceptual and implementation knowledge to better understand Priority Queues and Deques as a whole, which you'll then build on in Part 2 of the course series.

How You'll Learn

This course is 5 weeks long, and you're expected to spend 9–10 hours per week learning. You will watch a series of short videos explaining the concepts interspersed with interactive exercises to check your understanding. You will also have a visualization exploratory lab to dive deeper into understanding the data structures and algorithms.

Institution	Georgia Institute of Technology
Provider	edX
Instructor	Mary Hudachek-Buswell
Level	Intermediate
Workload	50 hours total
Enrollments	13K

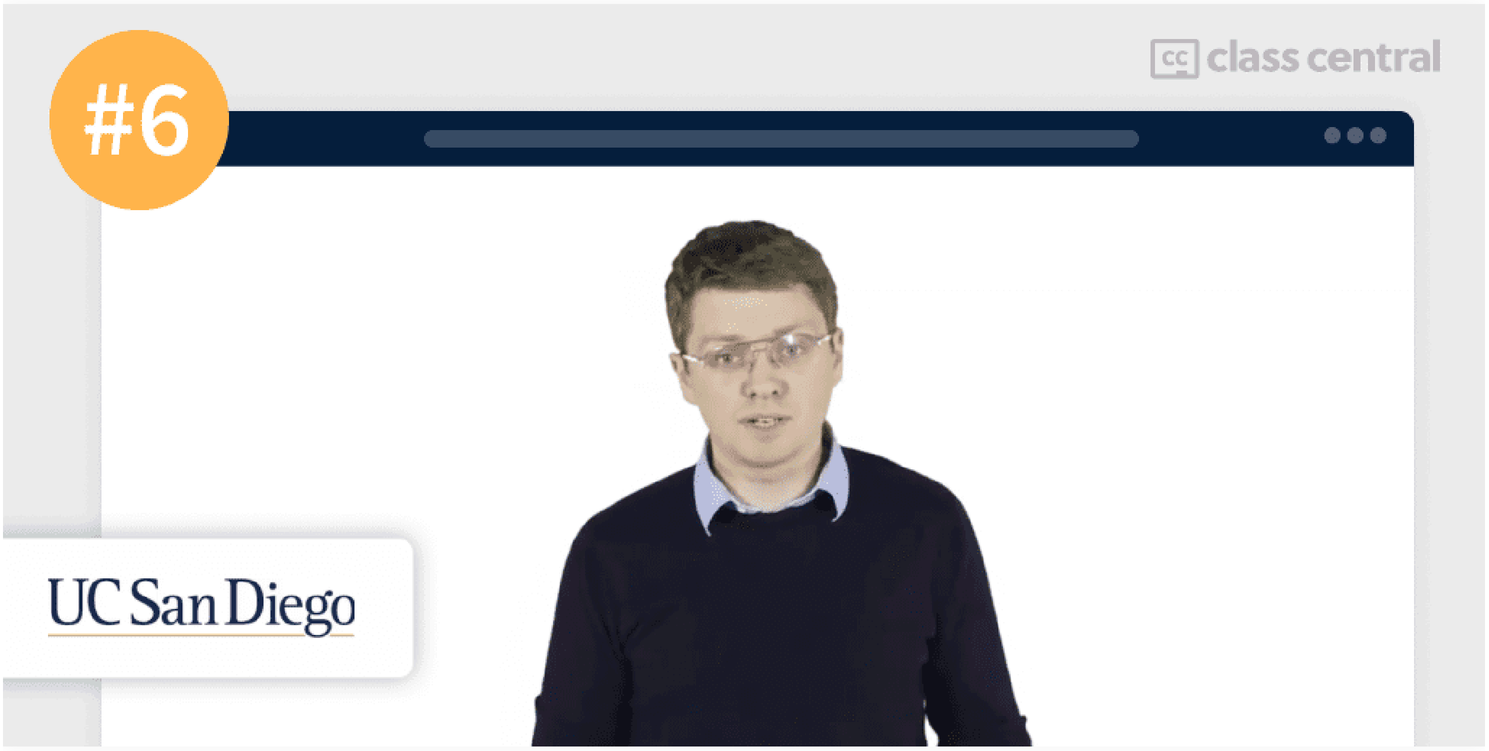
Certificate	Paid

Fun Facts

- Mary Hudachek-Buswell is a professor in the Computer Science department at Georgia Institute of Technology.
- It is one of the 100 most popular free online courses in 2022.
- *Data Structures & Algorithms I* is part of the Data Structures and Algorithms Professional Certificate.

If you're interested in this course, you can find more information about the course and how to enroll here.

6. Algorithmic Toolbox (University of California, San Diego)



Alexander Kulikov, course co-instructor

Offered by the University of California, San Diego, *Algorithmic Toolbox* covers the basic algorithmic techniques and ideas for computational problems that come up frequently in everyday applications. It's super hands on: you'll implement a ton of algorithms.

You will learn how to sort data and how it helps for searching, how to break a large problem into pieces and solve them recursively, when it makes sense to proceed greedily, and how dynamic programming is used in genomic studies. By practicing solving computational problems, designing new algorithms, and implementing efficient solutions, you'll become a better programmer.

To take this course, you'll need basic knowledge of any programming language as well as some topics in discrete mathematics including proof by induction and proof by contradiction.

What You'll Learn

This course covers sorting and searching, divide and conquer, greedy algorithms, and dynamic programming.

Starting with the first module, you'll go over an overview of where algorithms and data structures are used and tackle a few sample programming challenges, which will be abundant throughout the course.

Then, the second module introduces you to algorithms. Programs based on efficient algorithms can solve the same problem billions of times faster than programs based on naïve algorithms. In this module, you'll learn how to compare various algorithms and select the most efficient ones by estimating the running time and memory needs of an algorithm without even implementing it!

In the third module you'll learn about a seemingly naïve yet powerful class of algorithms called greedy algorithms. Although greedy algorithms can be extremely useful, they only work in certain situations. Hence, before using this sort of algorithm, it is important to prove that a greedy algorithm always produces an optimal solution. You'll gain intuition for building greedy algorithms by creating a program for changing money optimally.

The fourth module is all about Divide and Conquer. Based on this technique, you'll learn how to search huge databases millions of times faster than using naïve linear search. You will even learn that the standard way to multiply numbers is far from being the fastest! After that, you'll design two efficient algorithms (merge sort and quicksort) and apply them in a program that searches through huge lists, and finds a majority element. Finally, you'll prove that no other algorithm can sort faster!

Lastly, the fifth and sixth modules cover dynamic programming, a powerful algorithmic technique for solving many optimization problems. It turns out that dynamic programming solves many problems that are hard to solve with other methods like greedy algorithms or the divide-and-conquer strategy. Countless are the applications of dynamic programming: from maximizing the advertisement revenue of a TV station, to searching for similar Internet pages, to gene finding — you name it! You'll learn and apply dynamic programming to implement efficient programs.

How You'll Learn

This course is 6 weeks long, with each week taking approximately 4–8 hours of study. The course supplies video lectures and additional resources for you to learn from. Additionally, each week comes with multiple programming assignments for you to test your knowledge, if you are paying for the certificate.

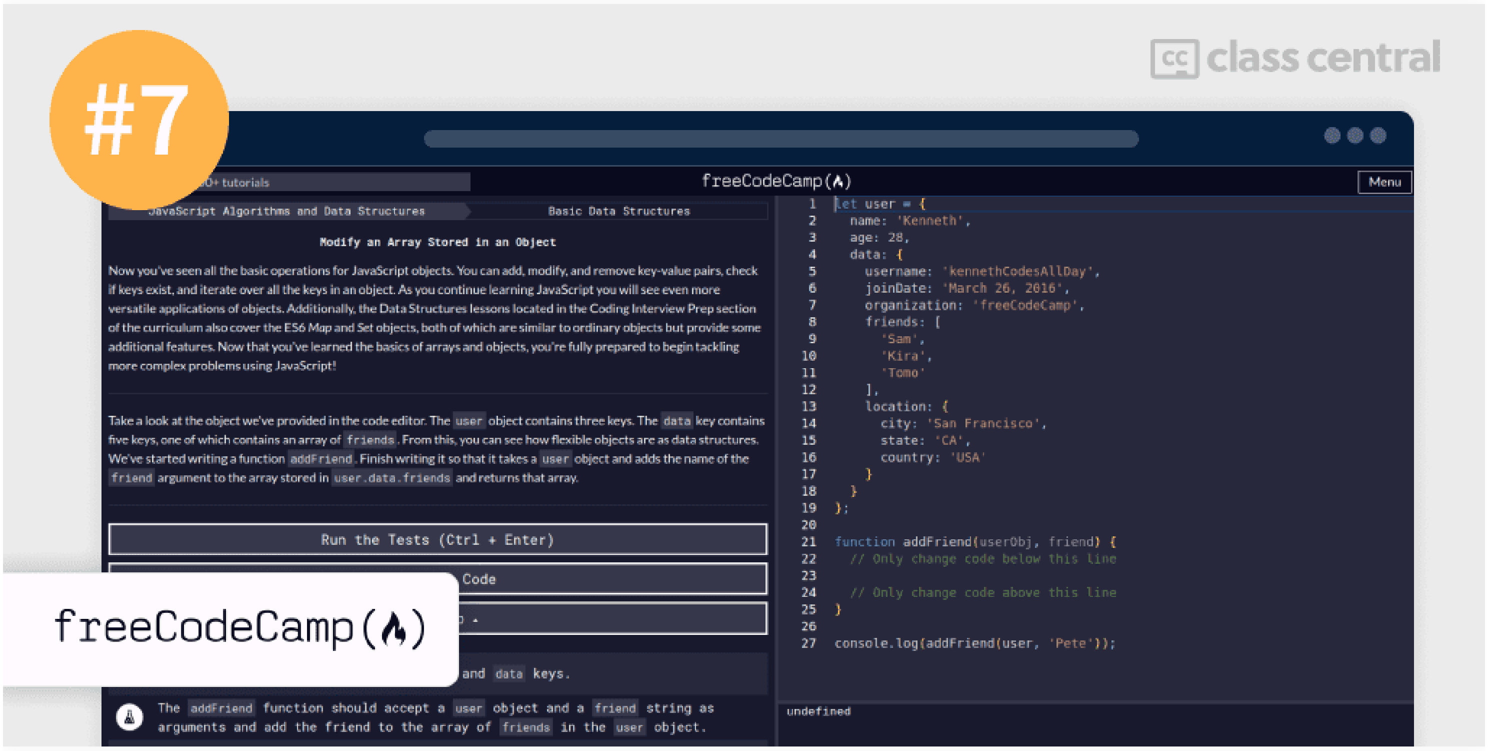
Institution	University of California, San Diego
Provider	Coursera
Instructors	Michael Levin, Daniel M Kane, Alexander S. Kulikov, Pavel Pevzner and Neil Rhodes
Level	Intermediate
Workload	40 hours total
Enrollments	431K
Rating	4.6 / 5.0 (12K)
Certificate	Paid

Fun Facts

- This course comes with a companion book that contains many solutions (including Python code) and hints for the problems in the course.
- The course is language agnostic and very hands on: you will implement a lot of algorithms yourself.
- It is the first course out of six in the Data Structures and Algorithms Specialization.

If you’re interested in this course, you can find more information about the course and how to enroll here.

7. JavaScript Algorithms and Data Structures (freeCodeCamp)



Modify an Array Stored in an Object

Now you've seen all the basic operations for JavaScript objects. You can add, modify, and remove key-value pairs, check if keys exist, and iterate over all the keys in an object. As you continue learning JavaScript you will see even more versatile applications of objects. Additionally, the Data Structures lessons located in the Coding Interview Prep section of the curriculum also cover the ES6 Map and Set objects, both of which are similar to ordinary objects but provide some additional features. Now that you've learned the basics of arrays and objects, you're fully prepared to begin tackling more complex problems using JavaScript!

Take a look at the object we've provided in the code editor. The `user` object contains three keys. The `data` key contains five keys, one of which contains an array of `friends`. From this, you can see how flexible objects are as data structures. We've started writing a function `addFriend`. Finish writing it so that it takes a `user` object and adds the name of the `friend` argument to the array stored in `user.data.friends` and returns that array.

and `data` keys.

The `addFriend` function should accept a `user` object and a `friend` string as arguments and add the friend to the array of `friends` in the `user` object.

```
let user = {
  name: 'Kenneth',
  age: 28,
  data: {
    username: 'kennethCodesAllDay',
    joinDate: 'March 26, 2016',
    organization: 'freeCodeCamp',
    friends: [
      'Sam',
      'Kira',
      'Tomo'
    ],
    location: {
      city: 'San Francisco',
      state: 'CA',
      country: 'USA'
    }
  }
};

function addFriend(userObj, friend) {
  // Only change code below this line

  // Only change code above this line
}

console.log(addFriend(user, 'Pete'));
```

```
undefined
```

freeCodeCamp in-browser code editor

Although *JavaScript Algorithms and Data Structures* is not strictly a DSA course, it does teach you the fundamentals of it. Hence it is also great for anyone interested in web development.

While HTML and CSS control the content and styling of a page, JavaScript is used to make it interactive. In the JavaScript Algorithm and Data Structures Certification, you’ll learn the fundamentals of JavaScript including variables, arrays, objects, loops, and functions.

Once you have the fundamentals down, you’ll apply that knowledge by creating algorithms to manipulate strings, factorialize numbers, and even calculate the orbit of the International Space Station.

Along the way, you’ll also learn two important programming styles or paradigms: Object Oriented Programing (OOP), and Functional Programing (FP).

So this course is a great 2-in-1 option: you'll learn both programming and algorithms/data structures.

What You'll Learn

The course covers nine topics: Basic JavaScript, ES6, Regular Expressions, Debugging, Basic Data Structures, Basic Algorithm Scripting, Object Oriented Programming, Functional Programming, and Intermediate Algorithm Scripting.

The first few sections of the course teach the fundamentals of JavaScript programming like arrays, objects, functions, loops, if/else statements, and more. It also teaches the ES6 features of the language like arrow functions, destructuring, classes, promises, and modules. You'll also learn how to match text patterns with Regular Expressions, along with learning how to debug your code using the JavaScript console.

Then, you'll move to the brunt of the course: Data Structures and Algorithms. You'll learn more about the differences between arrays and objects and which to use in different situations, as well as how to use methods like `splice()` and `Object.keys()` to access and manipulate data.

You'll then learn the fundamentals of algorithmic thinking by writing algorithms — series of step-by-step instructions to accomplish a goal — that do everything from converting temperatures to handling complex 2D arrays.

Afterwards, you'll study two programming paradigms: Object Oriented Programming and Functional Programming. In OOP, objects and classes organize code to describe things and what they can do, whereas in Functional Programming code is organized into smaller, basic functions that can be combined to build complex programs. Your knowledge of both paradigms will allow you to get the best of both worlds by writing more advanced programs like summing all primes or converting plain text to Pig Latin.

The course ends with five projects to put your JavaScript skills to the test. You'll build a palindrome checker, a roman numeral converter, caesars cipher encrypter, telephone number validator, and a cash register program.

How You'll Learn

This course is 100–200 hours long. The course is very hands-on: you'll write code from beginning to end. The most rewarding aspect of this course is the completion of the five projects needed to earn a free certificate, which you'll do all on your own.

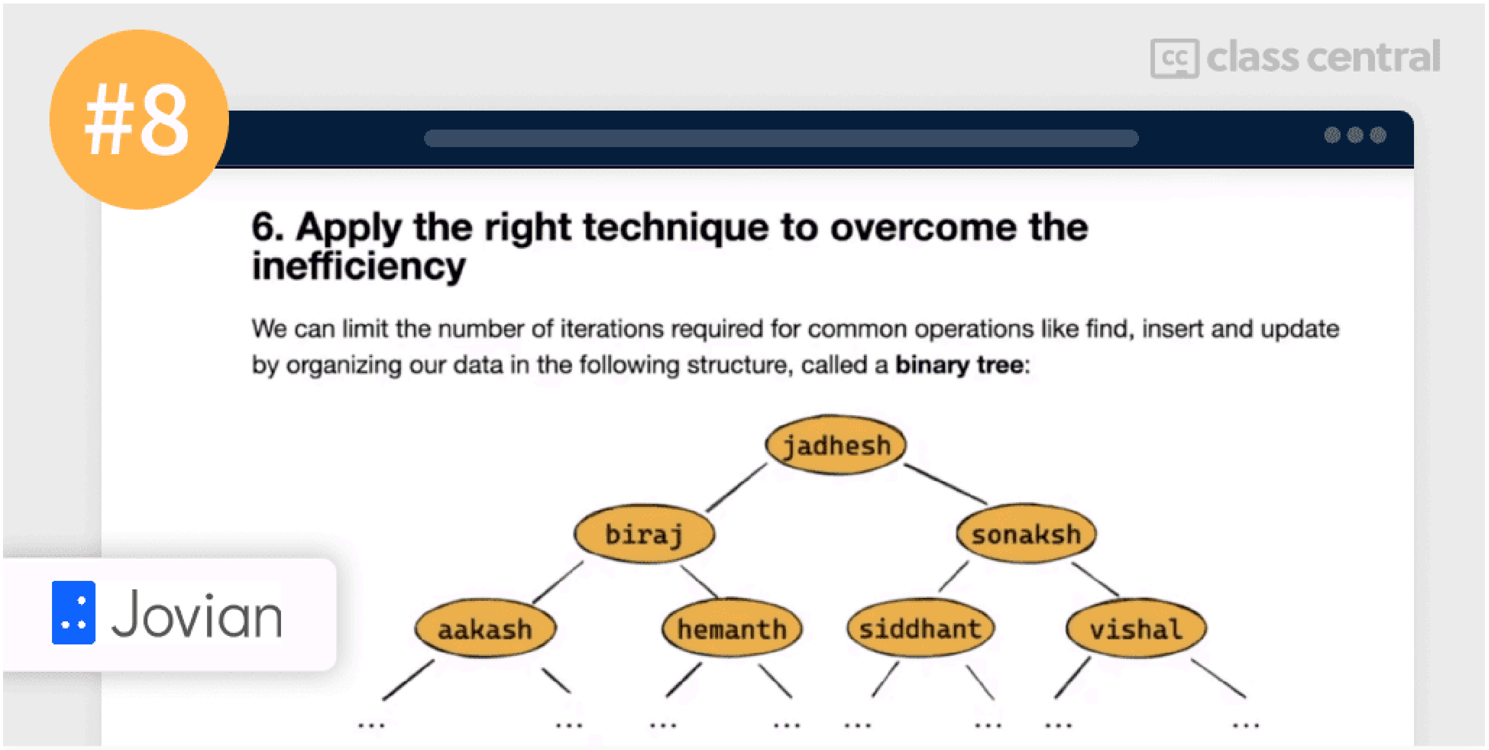
Institution	freeCodeCamp
Level	Beginner
Workload	100–200 hours total
Certificate	Free

Fun Facts

- The course has hundreds of bookmarks on Class Central.
- freeCodeCamp is a nonprofit that helps people learn to code for free. They offer 10 courses with free certification!
- freeCodeCamp's Youtube Channel has more than 5M subscribers and 1200+ videos on a wide variety of programming and computer science topics.
- If you'd like to support freeCodeCamp's mission of providing free education, you can make a donation here.

If you're interested in this course, you can find more information about the course and how to enroll here.

8. Data Structures and Algorithms in Python (Jovian)



6. Apply the right technique to overcome the inefficiency

We can limit the number of iterations required for common operations like find, insert and update by organizing our data in the following structure, called a **binary tree**:

Binary tree

Data Structures and Algorithms in Python is a beginner-friendly course that introduces you to common data structures (linked lists, stacks, queues, graphs) and algorithms (search, sorting, recursion, dynamic programming) in Python. By the end of the course, you'll be prepared to tackle coding interviews and assessment.

The course is offered by Jovian on the freeCodeCamp YouTube Channel.

Before taking this course, you'll need to know basic programming in Python as well as some highschool mathematics.

What You'll Learn

The course first introduces you to Big O notation — the way computer scientists measure the complexity of an algorithm. You'll then implement and compare the complexity of Binary Search and Linear Search, before learning about implementing linked lists using Python classes. Then, you'll study binary trees and how to traverse them, as well as the different types of binary trees and common operations.

Next, you'll learn about hash tables with Python dictionaries. You'll learn how to handle collisions using linear probing, and then learn how to replicate the Python dictionary data structure.

Knowing the differences between sorting algorithms can help you speed up your code. For example, you'll study and analyze bubble sort, insertion sort, merge sort, and quick sort. Using a strategy known as divide and conquer, you'll learn how to optimize polynomial multiplication and other algorithms. Lastly, you'll also learn how to analyze time and space complexity.

Dynamic programming is also a powerful method for optimizing your code. By tying in key concepts like recursion and memoization, you'll be able to apply them to a variety of problems, including the well-known knapsack problem.

The course then introduces you to graphs, trees, and adjacency lists which are used everywhere in computer science. You'll learn about breadth-first search and depth-first search, and find the shortest path between two nodes in a graph.

The final section of the course is all about teaching you how to think effectively as a programmer, as well as preparing and advising you for cracking coding interviews.

How You'll Learn

This course is 12.5 hours long. The contents of the course are taught in a video lecture format, but you'll be provided with plenty of Jupyter Notebooks to practice coding along with the lecturer.

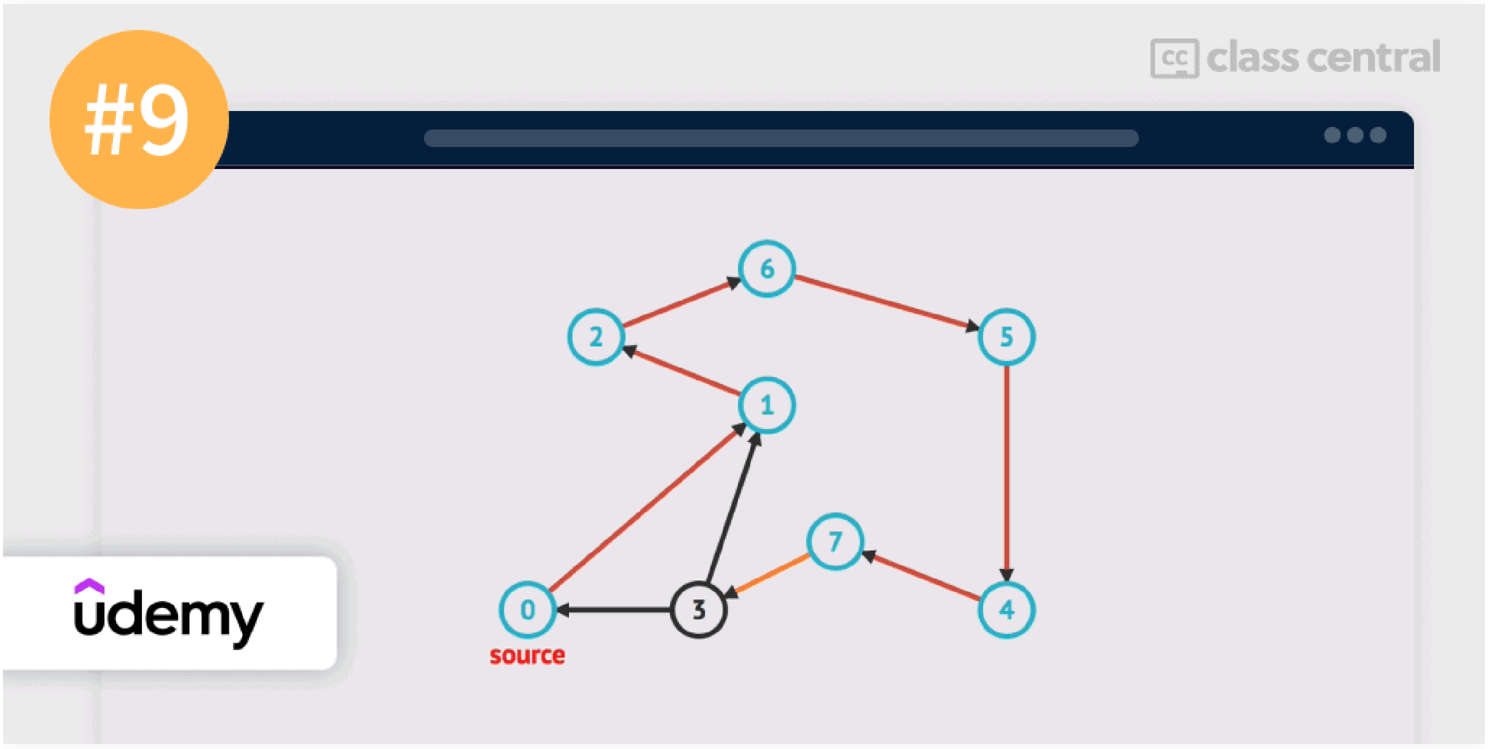
Institution	Jovian
Provider	freeCodeCamp
Instructor	Aakash N C
Level	Beginner
Workload	13 hours total
Views	626K
Likes	21K
Certificate	Free

Fun Facts

- This course is also available at Data Structures and Algorithms in Python.
- Jovian also offers many other Python courses related to Data Science, like Data Analysis with Python and Machine Learning with Python.

If you're interested in this course, you can find more information about the course and how to enroll here.

9. Master the Coding Interview: Data Structures + Algorithms (Udemy)



Graph traversal

If you're a self-taught developer or anyone anxious about facing a technical interview, this language-agnostic course might be what you're looking for.

Master the Coding Interview: Data Structures + Algorithms will help you cut through the technical interview process like a ninja.

Not only will it teach you about the fundamentals of data structures and algorithms and how to ace the coding questions given by leading companies like FAANG, it will also teach you the soft skills needed to increase your chances of getting accepted.

Another thing I like about this course is its focus on community. You'll have access to their private Discord server where you can get help, chat with other learners and mentors, and most importantly, keep yourself accountable.

What You'll Learn

This course has two sides: technical and non-technical.

For the technical side, you'll start by learning what makes code good. Any competent developer can write code, but what separates the best code from the rest? Enter time-complexity and space-complexity. You'll learn how to calculate how long your code might run and also how much space it might take up while running with the Big-O notation.

Once you have that sorted, you'll move on to data structures. A data structure is analogous to a container. You have many different containers like a cupboard, a fridge, a backpack, or a purse, but you wouldn't store ice cream in a cupboard nor money in a fridge! Hence, each data structure has its own advantages that makes it better suited for one thing than another.

You'll be taught some of the most common and important data structures, including arrays, hash tables, linked lists, stacks & queues, trees, and graphs. You'll not only learn the theory behind each of them but also how and when to implement them in your code.

Next, you'll study algorithms. Algorithms are steps in a process that we take to perform a desired action with computers. Combining algorithms with data structures allows us to achieve some remarkable results!

After an introduction to recursion and having written some recursive algorithms yourself, you'll study sorting algorithms and searching algorithms. Sorting and searching is an important process for companies dealing with millions of items like Amazon and it is very computationally expensive.

You'll learn bubble sort, selection sort, insertion sort, merge sort, quick sort, heap sort, and radix sort. You'll additionally learn about linear and binary search, as well as compare the differences between breadth-first search and depth-first search. Another thing you'll learn is dynamic programming, which is a technique that uses cache to optimize code.

For the non-technical side of the course, you'll learn how to get more interviews by building a portfolio, become more confident and prepared for your next coding interview, and also how to professionally handle offers and negotiate raises.

How You'll Learn

Although this course is 20 hours long, the course instructor recommends that you take at least a month to complete it. You'll mainly learn from watching the course videos, and also by putting theory into practice by coding along with the course and checking your code with the provided solutions.

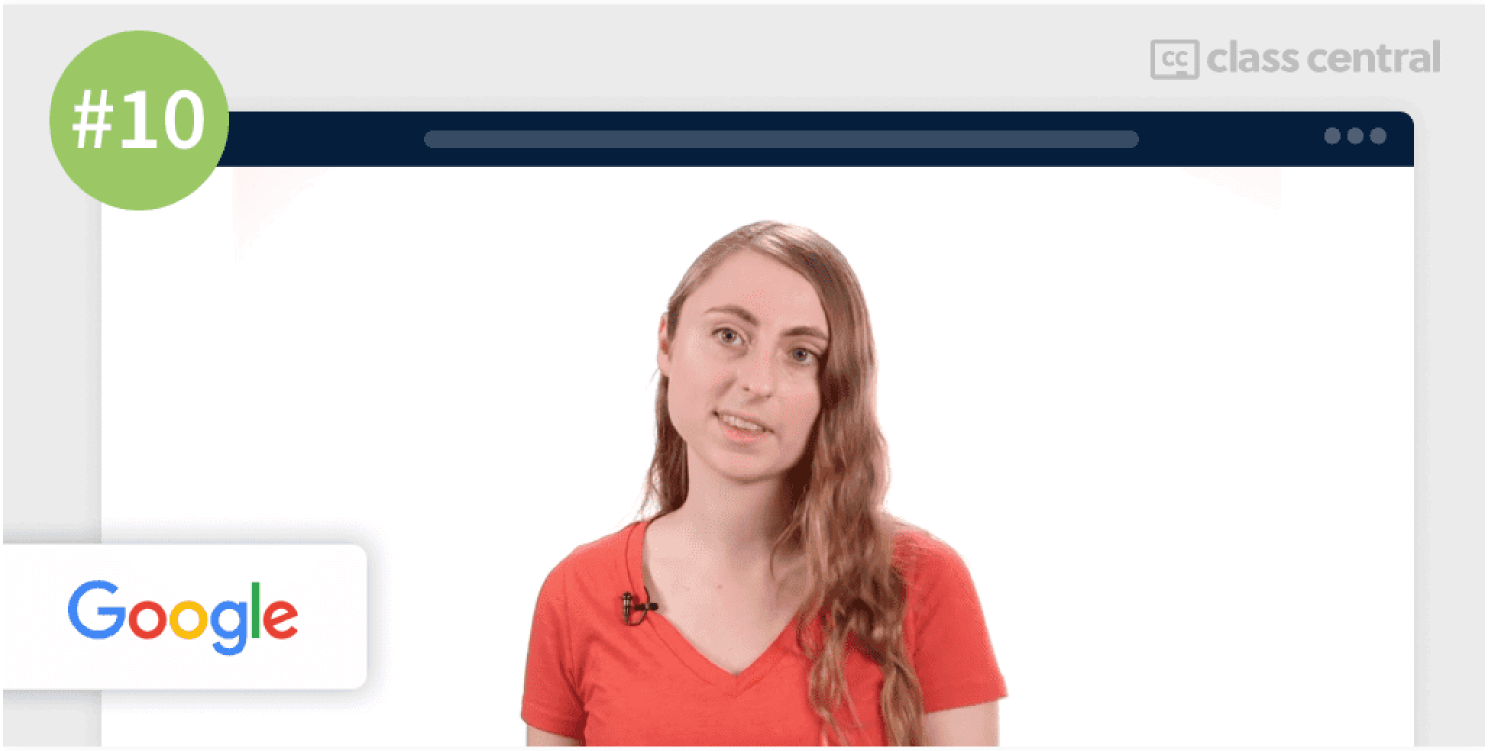
Provider	Udemy
Instructor	Andrei Neagoie
Level	Intermediate
Workload	20 hours total
Enrollments	143K
Rating	4.7 / 5.0 (23K)
Certificate	Paid

Fun Facts

- The course has an extra section on how to contribute to open source.
- Andrei is the instructor of some of the highest rated programming and technical courses online and is the founder of ZTM academy.
- He is also a self-taught programmer, and knows the struggles of being one.

If you're interested in this course, you can find more information about the course and how to enroll here.

10. Intro to Data Structures and Algorithms (Google)



Brynn Claypoole, course co-instructor

If you're preparing for a technical interview but have either never taken a course on data structures or algorithms, or need a refresher, then this course is ideal for you.

Offered by Google on Udacity, *Intro to Data Structures and Algorithms* will introduce you to common data structures and algorithms in Python to prepare you for technical interviews. You'll review frequently-asked technical interview questions and learn how to structure and explain your responses.

Additionally, you'll practice mock interviews to get specific recommendations for improvement. By the end of the course, you'll be ready for anything technical interviewers might throw at you.

To take this course, you'll need to be comfortable with algebra and coding in Python.

Note: this course uses Python 2 and not the latest version, Python 3, but the concepts are still relevant.

What You'll Learn

The course begins with an introduction to efficiency. You'll define what efficiency means when applied to algorithms, as well as explain the Big O notation commonly used to describe efficiency.

Then, you'll learn about list-based collections. You'll see definitions and examples of list-based data structures, like arrays, linked lists, stacks, and queues. After examining the efficiency of common list methods using what you've been taught, you'll practice manipulating these data structures.

Now that you have list-based data structures, the next thing you'll want to do is to know how to search and sort through them. You'll learn how to implement several algorithms (with recursion) like binary search, bubble sort, merge sort, and quick sort, as well as tell how efficient they are.

The course then introduces you to maps and hash functions. You'll understand and learn how to apply the concepts of sets, maps (dictionaries), and hashing to real-world problems.

Next, you'll learn about trees and paths, their concepts and terminology. You'll Investigate common tree types, such as binary search trees, heaps, and self-balancing trees and learn how to traverse and manipulate them. The same will be done with graphs, where you'll learn about their representations, properties, traversals, and paths, before analyzing their efficiency.

Nearing the end of the course, you'll discuss a few famous computer science problems like the Traveling Salesman Problem. You'll learn how brute-force, greedy, and dynamic programming are used to solve them.

Lastly, you'll learn about technical interviewing tips. You'll understand how to answer technical interview questions leveraging the concepts taught in this course. You'll gain some practice using Pramp to meet with other students for mock interviews.

How You'll Learn

This course runs for 4 weeks. Throughout the course, you'll learn concepts through video tutorials and watch experienced engineers explain their reasoning as they work their way through algorithmic problems.

Regarding assignments, there are plenty of practice coding exercises that you can work on to drill what you've been taught.

Institution	Google
Provider	Udacity
Instructors	Horatio Thomas and Brynn Claypoole
Level	Intermediate
Workload	4 weeks long
Certificate	None

Fun Facts

- Horatio is Student Experience Lead at Android while Brynn is a Lead Data Analyst at Udacity.

If you're interested in this course, you can find more information about the course and how to enroll here.

Elham Nazif
Part-time content writer, full-time computer science student.
More articles from Elham Nazif

Related

10 Best Data Science Courses to Take in 2024

10 Best Free Programming Courses to Take in 2024

10 Best Applied AI & ML Courses to Take in 2024

10 Best TensorFlow Courses to Take in 2024

About Class Central

Comments 6

Jim
4/18/2022 at 10:17pm

@Elham Thanks for your detailed summaries of every course! I'm excited by your series, because it expands on Manoel's analysis of "600+ Free Computer Science Courses from World's Top 50 Universities". (While I appreciate free courses as much as I still can, I also know that "free" no longer means what it used to mean in terms of content access.)

Unlike your programming article that was also limited to free courses, this article describes each course's content to let readers decide whether a course is worth its tuition. (Given two courses with similar content and comparable in quality though, tuition or lack thereof can certainly be a deciding factor in ranking the two.) I'm curious — why doesn't the second course in the specialization of the top-ranked course in your programming article rank, at all, in this article?

Rather than the course that Dhawal reviewed, your #2 pick doesn't have any reviews yet. So why do you prefer that pick over its sister course at https://www.classcentral.com/course/algorithms-divide-conquer-374 ? In the future, I'm hoping to see your series delve into the best courses within specific categories (like categorized by @Manoel) of a data science curriculum.

Reply

Elham Nazif
4/19/2022 at 11:37pm

Hi Jim, glad to hear my article is helpful!

Could you elaborate more on what you mean by 'specific categories of a data science curriculum'?

To answer your first question,
If you're referring to Python Data Structures by University of Michigan, the reason why I didn't include that is because it touches more on Python data structures (list, dictionaries, tuples) than data structures in general (heaps, linked-list, trees), and it does not teach algorithms.

To answer your second question,
I didn't pick the Coursera specialization because the Best Courses Guide reviews courses, not specializations. Comparing the first course in the edX professional certificate and the Coursera specialization, the edX courses covers more content in one course (split into 2 parts) than the Coursera courses (split into 4 parts) although both courses are part of the same curriculum. So I picked the edX course instead.

Reply

Jim
4/20/2022 at 11:44am

I see @Elham; I thought that workload could have been a deciding factor in your #2 pick. But I was unaware its content is equivalent to those of two courses in Stanford's "Algorithms" specialization that's offered by Coursera. (Since edX doesn't yet offer a specialization that includes your pick, I mistakenly assumed that your pick's content was equivalent to the single course linked in my original comment and that edX will catch-up to Coursera by adding the other courses in the "Algorithms" specialization.)

P.S. for data science, @Manoel categorized his list of courses under:
Data Analysis
Big Data
Data Visualization
Data Mining

Reply

Elham Nazif
4/21/2022 at 8:51am

Those data science topics look promising to write about, thanks for the suggestion!

Reply

Jim
4/21/2022 at 5:11pm

@Elham, I'd also like to have a category for Data Preparation. There are several languages that are popular for extracting and cleaning data. Thanks!

Reply

TranHuy
3/31/2023 at 11:31am

Is there any course using C++

Reply

Leave a reply

Your email address will not be published. All comments go through moderation, so your comment won't display immediately.

Comment

Name *(Required)*

Email *(Required)*

Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Browse our catalog

Discover thousands of free online courses from top universities around the world like MIT, Stanford, and Harvard.

Computer Science
13,168 courses

Artificial Intelligence Algorithms and Data Structures Internet of Things Information Technology Computer Networking Machine Learning DevOps Deep Learning Cryptography Quantum Computing Human-Computer Interaction (HCI) Distributed Systems Blockchain Development Operating Systems Computer Graphics Automata Theory Compilers SCADA Mainframe Digital Image Processing CSS Animation Morph Transition CSS-in-JS

Business
21,417 courses

Management & Leadership Finance Entrepreneurship Marketing Strategic Management Industry Specific Business Intelligence Accounting Human Resources Project Management Sales Design Thinking Business Software Customer Service Nonprofit Management Innovation

Operations Management Corporate Governance Business Plan Business Proposal
Management Consulting Business Math

Humanities
8,301 courses

History Literature Language Learning Grammar & Writing Philosophy Religion ESL Culture
Sports Journalism Ethics Linguistics Food Library Science Reading Crisis Management
Games Emergency Management Language Arts

Data Science
4,791 courses

Bioinformatics Big Data Data Mining Data Analysis Data Visualization Jupyter Notebooks
Process Mining Stata Text Mining Data Bias Topological Data Analysis

Personal Development
5,702 courses

Communication Skills Career Development Self Improvement Presentation Skills Resilience
Charisma Self-Acceptance Mental Toughness Self-Doubt Management Kindness Self-love
Personal Empowerment Habit Tracking

Art & Design
20,638 courses

Music Digital Media Visual Arts Design & Creativity Art Therapy Art Composition

Browse all subjects

/ The Report

Browse by subject

Computer Science

Psychology

Cybersecurity

Health

Law

Accounting

Web Development

Browse by provider

Coursera

edX

FutureLearn

Udacity

Swayam

Udemy

freeCodeCamp

Browse by university

Harvard

Stanford

Georgia Tech

University of Michigan

Purdue University

Duke University

IIT Madras

Browse by institution

Google

Microsoft

IBM

Amazon

Linux Foundation

British Council

Salesforce

Rankings

Best Online Courses of All Time

Best Online Courses of the Year (2022 Edition)

Most Popular Courses of All Time

Most Popular Courses of the Year (2022 Edition)

250 Top FREE Coursera Courses of All Time

100 Top FREE edX Courses of All Time

250 Top Udemy Courses of All Time

by Class Central

The "New Normal" that Wasn't

DDoS Attack on Class Central

500+ Online Degrees in India

Harvard's CS50 Free Certificate Guide

How Open University Works

Free Courses and Certificates

600 Free Google Certificates

9000 Free Courses from Tech Giants

1700 Free Coursera Courses

Ivy League Online Courses

175+ Free Writing Online Courses

About Class Central

Class Central aggregates courses from many providers to help you find the best courses on almost any subject, wherever they exist. *The Report* by Class Central, is your source for the latest news and trends in online education.