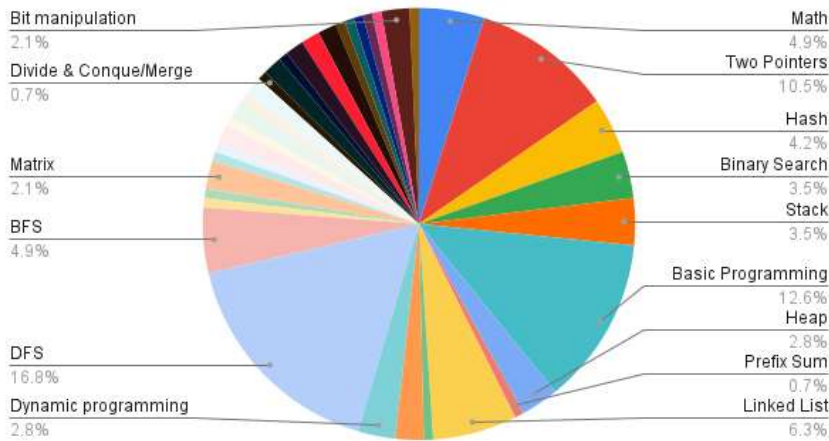


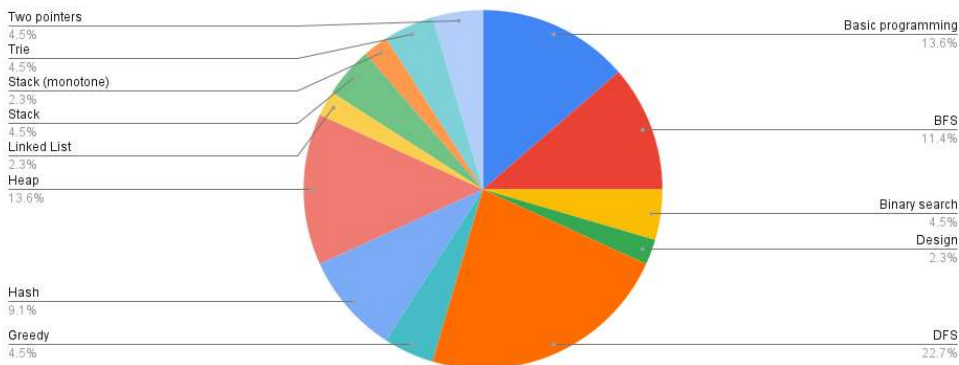
Top Patterns to Conquer the Technical Coding Interview

Since the goal of AlgoMonster is to help you get a job in the shortest amount of time possible in a **data-driven** way, we compiled datasets of tech interview problems and broke them down by patterns. This way we can figure out the **most frequently tested** and **highest return on investment (ROI)** area to focus on. We'll look at the breakdown for top problems overall as well as company specific breakdowns.

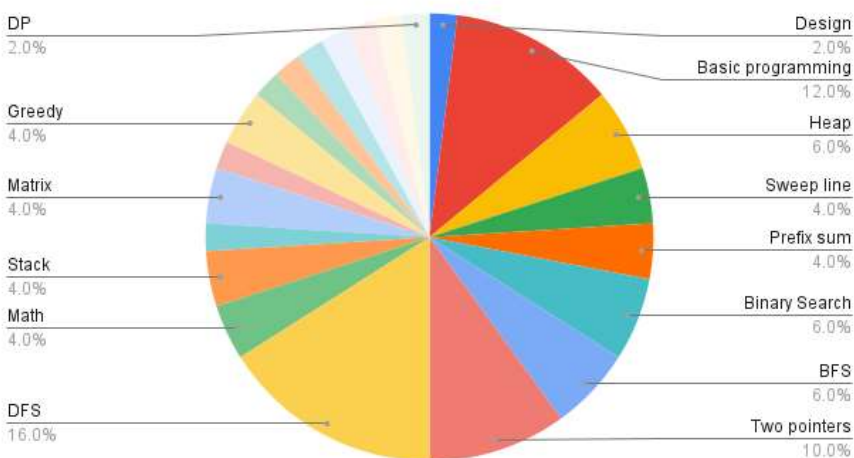
Top problems overall



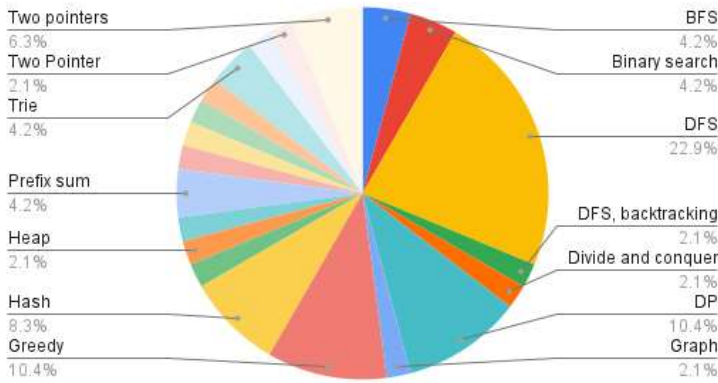
Amazon



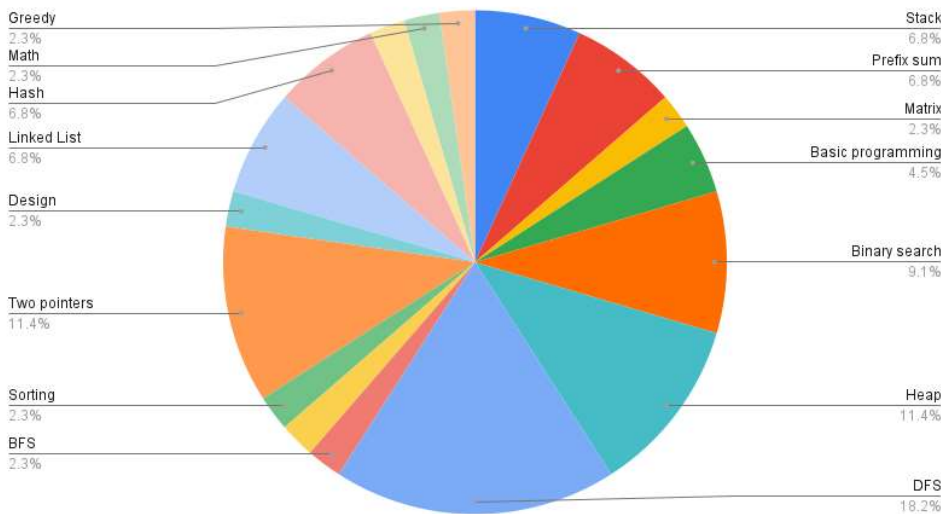
Facebook



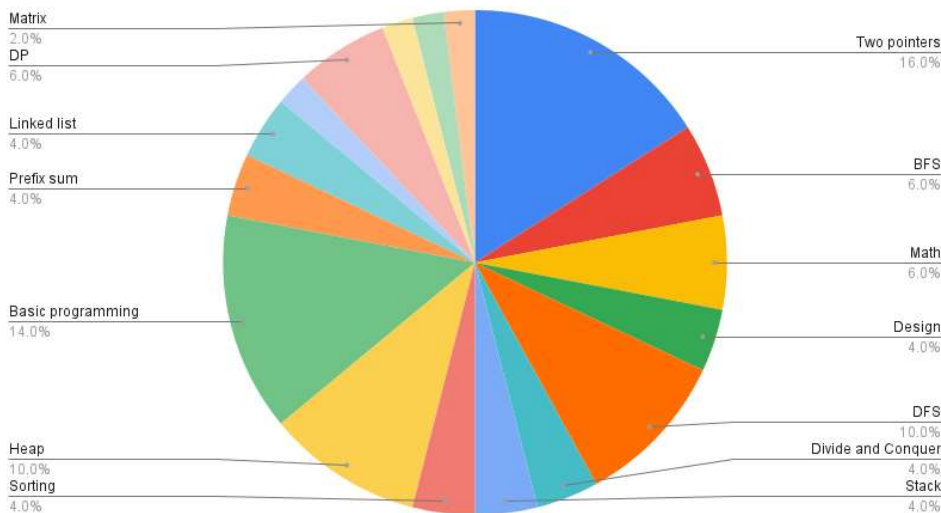
Google



Expedia



Airbnb



Trends

Highest ROI: DFS, BFS, Two pointers, Binary Search

[Depth-first search](#)(DFS), [breadth-first search](#)(BFS), [binary search](#) and [two pointers](#), make up a good portion of all interview problems. DFS in particular can be used to solve a wide range of problems from tree to graph to combinatorial problems and is very useful in tech interviews. We have a detailed explanation on each of these topics.

Linked list, stack, and queue high ROI

These problems get asked a lot but there aren't many variations. For example, linked list has several classic problems such as reversing linked list, find middle nodes, and linked list cycles detection that get asked over and over.

Priority Queue/heap medium ROI

[Priority queue and heap](#) show up **surprisingly more often than most would think**. The classic problems include the median of a data stream, [k closest points](#).

Greedy, Dynamic programming

[Dynamic programming](#) (DP) is hard to summarize and does not appear frequently (unless you are interviewing Google whose engineers like to ask DP problems). There are a few classic problems you may want to know such as [house robber](#) and [robot paths](#). Otherwise, you should probably focus on higher ROI areas if you are short on time. Also if you can't figure out a dynamic programming solution, you can always do DFS + memoization which does the same thing.

Each greedy problem is different and it's hard to summarize a pattern, and the correctness of your solution often requires rigorous mathematical proof that is hard to learn in a short time. Therefore, we consider the greedy problem to be low ROI.

Trie, Union Find medium-low ROI

These don't appear often but do show up. Consider them of secondary priority.

Basic programming

You may have noticed there is a category called "basic programming". We use this to denote problems that don't fit into any category and are either fairly simple that you can practice yourself or **code-to-specifications** problems. This makes up a good portion of all interview problems. You don't need any prior knowledge other than a basic programming concept to solve them, and we don't cover them in this course since you can learn them and practice yourself.

What to study

We created an ROI table based on the above analysis for your reference.

Topic	Difficulty to learn	Return of Investment
Two Pointers	Medium	High
Tree BFS	Low	High
Graph BFS	Medium	High
Tree DFS	Medium	High
Graph DFS	Medium	Medium
Combinatorial DFS	Medium	Medium
Heap	Medium	Medium
Binary Search	Medium	Medium
Divide and Conquer	High	Low
Linked List	Low	High
Hash	Medium	High
Dynamic Programming	High	Low
Trie	Medium	Medium
Union Find	Medium	Low
Greedy	High	Low

Company-specific trends

Amazon

Amazon's interview problems haven't changed much since the Cracking the Coding Interview days. Two pointers, DFS, BFS make up half the problems. Since the existence of the [performance improvement plan](#), it's relatively easy to fire employees. So the interviews are not intended to be very difficult.

Facebook

Facebook likes to ask classic problems and is generally harder than Amazon. Word on the streets is the company cares more about bug-free coding than anything else. Two pointers, DFS and BFS still dominate. If you are interviewing Facebook it's a good idea to practice the classic problems we have in this course multiple times.

Google

Google has an internal policy of not using any problem that can be found on the internet for interviews. Therefore the engineers constantly reinvent new problems which also makes the interview more difficult (or easier depending on how you view it since invented problems are often as well-thought-out as classic problems). Some problems require more than one pattern, e.g. prefix sum + binary search, DFS + prefix sum. You have to be very familiar with the basic patterns we have in this course.

20th Century Blue Chips like Oracle, IBM, Expedia

These companies have a fixed set of problems and don't normally update them. If you master the patterns we teach in this course, you should be able to solve them with ease.

Unicorn Startups

Unicorn startups like Airbnb, Uber, Dropbox have a relatively stable problem bank compared to Google and the problems are harder than Amazon. Classic patterns like DFS, BFS, two pointers are still the favorite. You may need to know Trie and union-find, too.

A Note On Academic Algorithms

We use the term "academic algorithms" to mean algorithms that are taught in university textbooks and are not tested often if ever in real-world tech interviews according to our statistics. A good-enough rule of thumb is an algorithm that's named after a person(s) you can safely ignore.

Very rarely/Never used list:

- Minimal spanning tree: Kruskal's algorithm and Prim's algorithm
- Minimum cut: Ford–Fulkerson algorithm
- Shortest path in weight graphs: Bellman–Ford–Moore algorithm
- String search: Boyer–Moore algorithm

Knuth–Morris–Pratt (KMP) algorithm is useful in solving string problems but interviewers will NOT expect to know how to do this. Normally bug-free brute force coding is required instead of KMP.

Dijkstra's algorithm is useful in finding the shortest path between nodes in a weight graph. Weight graphs interview problems exist but are rare. It's good to have a high-level understanding of this algorithm since it uses a priority queue which gets asked relatively often.

[Robin Karp rolling hash algorithm](#) is sometimes useful in solving certain two-pointer problems.

Reference: you can find the data source [here](#).

Get started on the highest ROI patterns!

- [Binary search](#)
- [Breadth-first search](#)
- [Depth-first search](#)
- [Two pointers](#)
- [Priority Queue](#)

Go Premium


Mark as Completed

Next Up: Keyword to Algo Cheat Sheet

Nickname

Email (optional)

Reply...



Comment

Mike

2022-03-29 10:58

I want to learn a lot in short period of time and study after work

am MOD

2022-03-29 19:16

Focus on the "Core Patterns" first.

Reply

Reply

Arun

2022-02-16 10:17

I have Amazon and Microsoft interviews scheduled in 2 weeks. Please help me out what should I mainly focus on.

Reply

bottles

2022-01-17 21:49

the Google pie chart has slices for both "Two pointer" and "Two pointers" - should these be combined into one slice?

Reply

Alap

2021-11-17 22:05

So I have a Meta (Facebook) interview in 3 weeks on the dot and currently am not good at technical interviews. Should I just be cranking out and cramming the Facebook suggested topics like Two pointers, DFS and BFS? Please let me know!

Sam

2022-02-24 22:01

How'd the interview go?

Reply

stoof

2022-02-17 15:37

How did it go?

Reply

A4Awesome

2021-12-02 19:34

I'm in the exact same boat! Let us know how it goes and all the best. I hope to hear what others have to say on this.

Sid

2022-02-16 02:23

Folks, how did the interviews go? Any feedback/ suggestions on the prep? Thanks!

Reply

Reply

Reply

