## FAANG ROADMAP

Algorithms		System design		Behavioral		Mock interviews, paid courses	
Theory	<ol> <li>Algorithms course (part 1, 2). Robert Sedgewick, Coursera (https://www.coursera.org/learn/algorithms-part1)</li> <li>Introduction to Algorithms. Thomas Cormen         <ul> <li>http://e-maxx.ru</li> <li>https://algolist.ru</li> <li>https://github.com/PetarV-/Algorithms</li> <li>https://leetcode.com</li> <li>https://www.codewars.com</li> <li>https://codeforces.com</li> <li>https://icpcarchive.github.io or https://icpc.global/worldfinals/past-problems</li> </ul> </li> </ol>	1. Designing Data-Intensive Applications. Martin Kleppmar 2. System Design Interview (vol. 1, 2). Alex Xu	nn Theory Practise	<ol> <li>Software Engineering at Google. Titus W., T</li> <li>Amazon leadership principles</li> <li>~5 stories with STAR methodology</li> <li>Live mock interviews</li> </ol>	om M., Hyrum W.	<ol> <li>https://www.algoexpert.io</li> <li>https://www.pramp.com</li> <li>https://www.interviewbit.com</li> <li>https://bytebytego.com</li> <li>https://www.tryexponent.com and https://www.youtube.com/@tryexponent</li> </ol>	
	SQL/NoSQL/DBs  1. Transaction isolation level 2. CAP theorem 3. Replication/Partitioning/Sharding 4. Migration 5. Locks 6. Indexes (types, usage)	Big Data/Data Engineering  1. ETL pipelines 2. Airflow, Kafka, Spark 3. Hadoop	<ol> <li>Clean Architect</li> <li>Fundamentals ( Approach. Neal</li> <li>Software Archit</li> <li>Building Micros</li> <li>Microservices p</li> </ol>	e Architecture  ure. Robert C. Martin of Software Architecture: An Engineering Ford ecture: The Hard Parts. Neal Ford ervices. Sam Newman atterns. Chris Richardson oservices. Thomas Hunter	<ol> <li>Design Patterr</li> <li>Head First Des Sierra</li> <li>Refactoring. M</li> <li>Clean Code. Ro</li> </ol>	Patterns, refactoring  1. Design Patterns. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 2. Head First Design Patterns. Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra 3. Refactoring. Martin Fowler 4. Clean Code. Robert C. Martin 5. Code Complete. Steve McConnell	
2.	General  Competitive Programmer's Handbook. Antti Laaksonen Structure and Interpretation of Computer Programs. Harold Abelson, Gerald Jay Sussman <a href="https://sijinjoseph.com/programmer-competency-matrix">https://sijinjoseph.com/programmer-competency-matrix</a>	Math  1. Mathematics for Computer Science. Eric L., Tom L  2. Discrete mathematics for computing. Haggarty R.	b. Pro git. S	Tools  Jocumentation (https://git-scm.com/doc) Goott Chacon and Ben Straub (https://git-n/book/en/v2)		Technologies Docker Kubernetes	

DevOPS/SRE

2. Ansible

3. Jenkins

1. Continuous delivery

Computer architecture/Operating systems

1. Structured Computer Organization. Andrew Tanenbaum

2. Modern Operating Systems. Andrew Tanenbaum