Technical Interview Preparation

Suggested prepartion material to review:
[1] "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
[2] "Five Essential Phone Screen Questions" by Steve Yegge
[3] "Sample Google Interview Questions"
[4] "Hangout on Air: Candidate Coaching Session"
[5] "Programming Interviews Exposed: Secrets to Landing Your Next Job" by John Mongan, Noah Suojanen

What to expect from the interview:
Knowledge of computer science principles (data structures, algorithms, systems design, and Big O notation etc.) and how they can be used in your solutions. The main areas of preparation to succeed:

Algorithm Complexity: You need to know Big-O. If you struggle with basic big-O complexity analysis, then you are almost guaranteed not to get hired. To brush up on algorithms & data structures visit: *topcoder.com*

Coding: You should know at least one programming language really well (preferably Python, C++ or Java). Be sure to check out our Google code style guides.

Sorting: Know how to sort. Don't do bubble-sort. You should know the details of at least one n*log(n) sorting algorithm, preferably two (say, quicksort and merge sort). Merge sort can be highly useful in situations where quicksort is impractical, so it would be beneficial to take a look at it.

Hashtables*:* Arguably the single most important data structure known to mankind. Be able to implement one using only arrays in your favorite language, in about the space of one interview.

Trees: Know about trees; basic tree construction, traversal and manipulation algorithms. Familiarize yourself with binary trees, n-ary trees, and trie-trees. Be familiar with at least one type of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree, and know how it's implemented. Understand tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.

Graphs: Graphs are really important at Google. There are 3 basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list); familiarize yourself with each representation and its pros & cons. You should know the basic graph traversal algorithms: breadth-first search and depth-first search. Know their computational complexity, their tradeoffs, and how to implement them in real code. If you get a chance, try to study up on fancier algorithms, such as: Dijkstra and A*.

Other data structures: Study up on as many other data structures and algorithms as possible. Especially know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem, and be able to recognize them when an interviewer asks you them in disguise. Find out what NP-complete means.

Mathematics: Some interviewers ask basic discrete math questions. This is more prevalent at Google than at other companies because we are surrounded by counting problems, probability problems, and

other Discrete Math 101 situations. Spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of combinatorics and probability. Be familiar with n-choose-k problems and their ilk – the more the better.