Interview Preparation Tips

Tips on how to succeed:
At Google, we believe in collaboration and sharing ideas. Most importantly, you'll need more information from the interviewer to analyze & answer the question to its full extent.

* Its OK to question your interviewer.
* When asked to provide a solution, first define and framework the problem as you see it.
* If you don't understand - ask for help or clarification.
* If you need to assume something - verbally check it's a correct assumption!
* Describe how you want to tackle solving each part of the question.
* Always let your interviewer know what you are thinking as he/she will be as interested in your process of thought as your solution. Also, if you're stuck, they may provide hints if they know what you're doing.
* Finally, listen - don't miss a hint if your interviewer is trying to assist you!

Technical Preparation Tips:
The main areas software engineers should prepare to succeed at interview at Google:

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.

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 take a look at it.

Hashtables: Arguably the single most important data structure known to mankind. You absolutely should know how they work. 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: You should study up on as many other data structures and algorithms as possible. You should 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. You should be familiar with n-choose-k problems and their ilk – the more the better.

Operating Systems: Know about processes, threads and concurrency issues. Know about locks and mutexes and semaphores and monitors and how they work. Know about deadlock and livelock and how to avoid them. Know what resources a processes needs, and a thread needs, and how context switching works, and how it's initiated by the operating system and underlying hardware. Know a little about scheduling. The world is rapidly moving towards multi-core, so know the fundamentals of "modern" concurrency constructs.

Coding: You should know at least one programming language really well, and it should preferably be C++ or Java. C# is OK too, since it's pretty similar to Java. You will be expected to write some code in at least some of your interviews. You will be expected to know a fair amount of detail about your favorite programming language.

Final Tips:
***Get your algorithms straight (they may comprise up to a third of your interview). Visit: http://en.wikipedia.org/wiki/List_of_algorithm_general_topics and examine this list of algorithms: http://en.wikipedia.org/wiki/List_of_algorithms\ and data structures: http://en.wikipedia.org/wiki/List_of_data_structures Write out all the algorithms yourself from start to finish and make sure they're working.