Cracking The Coding Interview

powers of 2: 7, 8, 10, 16, 20, 30, 32, 40	128, 256, 1024(1 thousand, 1K), 65,536(64K), 1million(1M), 1billion(1G), 4G, 1trillion(1T)
Walking Through A Problem	7: Listen, Example, Brute Force, Optimize, Walk Through, Implement, Test
Optimize & Solve Techniques	1: Look for BUD, 2: DIY, 3: Simplify & Generalize, 4: Base Case & Build, 5: Data Structure Brainstorm
BUD	Bottleneck, Unnecessary work, Duplicated work (a way to approach optimizing: these are 3 of the most common things an algorithm can "waste" time doing)
BCR	Best Conceivable Runtime: best runtime you could *conceive* of a solution to the problem having. You can easily prove there is no way to beat the BCR.
Walking Through A Problem: Optimize (7)	1. Look for any UNUSED information 2. Use a FRESH EXAMPLE 3. Solve it INCORRECTLY 4. Make TIME VS. SPACE tradeoffs 5. PRECOMPUTE information 6. Use a HASH TABLE 7. Think about BCR (Best Conceivable Runtime)
Optimize & Solve Techniques: DIY	How would you do it on your own (without a computer)? E.g. find all permutations of a shorter string in a longer one 'abbc' in 'cbabadcbbabbcbabaabccbabc'
Object-Oriented Design: good first questions?	6 W's: Who What Where When How Why (e.g. 'design coffee maker' for massive restaurant or elderly couple at home?)
Object-Oriented Design: steps	4: Handle Ambiguity (6 W's), Define Core Objects (Classes), Analyze Relationships (Relations), Investigate Actions (Methods)
Implementation of Hash Tables	Array of Linked Lists. 1. Compute key's hash code (usually an int or long) 2. Map the hash code to an index in the array 3. There will be a linked list at that index, store the key and value in that linked list. Note that steps 1 and 2 may create collisions, which is why you need a linked list here.
Stack has what operations?	4: push, pop, peek (return top item but don't pop it), isEmpty
Queue has what operations?	4: add, remove, peek (return first item but don't remove), isEmpty
Tree definition	A Tree has one root node. A root node has zero or more child nodes. Each child node has zero or more child nodes.
What 3 types of Trees is it important not to confuse?	1. Tree 2. Binary Tree 3. Binary Search Tree: 1. Any number children 2. No more than 2 children 3. All left descendants <= n < all right descendants (specific equality rule may vary)
Balanced Tree definition	"not terribly unbalanced" - usually means something like there's not more than 1 level of difference between any two leaves
Complete Binary Tree	Every level of tree is filled, except last level, which is filled from left to right
Full Binary Tree	Every node has either zero or two children (i.e. no nodes have only one child)
Perfect Binary Tree	Both Full and Complete. All leaf nodes are at same level, and this level has the max number of nodes. Perfect Binary Trees are rare interviews and in the real world. They must have exactly 2**k - 1 nodes, where k is the number of levels

in-order traversal	left child, node, right child
	def in_order(node)
	  if (node != nil)
	  in_order(node.left)
	  visit(node)
	  in_order(node.right)
	  end
	end
	On a binary search tree, this visits the nodes in ascending order (hence the name)

pre-order traversal	node, left child, right child
	def pre_order(node)
	  if (node != nil)
	  visit(node)
	  in_order(node.left)
	  in_order(node.right)
	  end
	end

post-order traversal	left child, right child, node
	def post_order(node)
	  if (node != nil)
	  in_order(node.left)
	  in_order(node.right)
	  visit(node)
	  end
	end