HCS Bootcamp 1:
Technical Interview Workshop Small Group Problems

Harvard Computer Society

January 30, 2013

DISCLAIMER. I compiled these problems mainly through searching online and CS124. Your interview preparation should **not** be limited to this list. It is simply to give you a flavor of the kinds of problems you **might** see in an interview.

DIRECTIONS. For each of these problems, solve the problem in your most familiar language with the most efficient algorithm (in terms of time and space complexity), and state the time and space complexity of your algorithm. We will discuss the solutions in our small group and larger group if we have time.

1 Quickies

1. **Reverse a Linked List.** Given a pointer to the first node of a singly linked list, write a procedure to reverse the linked list. Do this recursively and iteratively. What is the space and time complexity of each implementation? Which way is better, and why?

2. **Valid Binary Search Trees.** A binary tree rooted at r is a binary search tree if for all nodes u in the left subtree of r, we have $u < r$, and for all nodes v in the right subtree of r, we have $r < v$. Given a binary tree, determine if the binary tree is a binary search tree.

3. **Needle in Haystack.** Given a string *needle* and string *haystack*, return $TRUE$ if *needle* is a substring of *haystack*, otherwise return $FALSE$.

4. `atoi` **(ASCII to Integer).** Implement the function `int atoi(char *str)` that when given a string `str`, returns the numeric value of this string. For example, `atoi("42") == 42`, `atoi("-35") == -35`. You may assume that `str` contains only numeric characters and possibly a leading negative sign.

2 Requires More Thought

1. **Two-Sum.** Given an array of n integers and a target integer t, return $TRUE$ if there exists $0 \leqslant i < j < n$ such that $a[i] + a[j] = t$, otherwise return FALSE.

2. **Three-Sum.** Given an array of n integers and a target integer t, return $TRUE$ if there exists $0 \leqslant i < j < k < n$ such that $a[i] + a[j] + a[k] = t$, otherwise return FALSE.

3. **Shuffle.** Given an array of n integers, write a function `shuffle` that when given the array, shuffles the array in place such that all $n!$ permutations of the n integers are equally likely. You may assume that you have a random number generator that when given a non-negative integer $i < 2^{32}$, can generate integers in the range $[0, i)$ such that any integer in this range is equally likely. Prove that your algorithm generates every permutation with equal probability.

4. **Maximal Subarray.** Given an array of n integers, determine the maximum sum that can be generated by a *continuous* subarray of the input array. For example:

$$\begin{aligned} 1, 2, 3, 4 &\rightarrow 1 + 2 + 3 + 4 = 10 \\ 2, 3, -1, -3 &\rightarrow 2 + 3 = 5 \\ -1, 5, 100, -1000 &\rightarrow 5 + 100 = 105 \\ -1, -2, -3, -4 &\rightarrow 0 \\ \textit{empty} &\rightarrow 0 \\ 1000, 2000, -1, 3000 &\rightarrow 1000 + 2000 - 1 + 3000 = 5999 \end{aligned}$$

5. **Stocks.** Given an array of n integers representing the price of a stock over the course of n days, determine the maximum profit you can make if you can buy and sell exactly 1 stock over these n days. Provide a divide & conquer algorithm, and a dynamic programming algorithm. Which is better? What is the time and space complexity for both solutions?

6. **LRU Cache.** A cache is a data structure that allows you to store a limited number of items and retrieve them very quickly. In the language of your choice, write down the interface (whatever an interface means in your language) for a cache. Now implement the cache using an LRU policy (this means that if the cache is full, the least recently used item is evicted to make space for the new item).