Array questions
Fighting the demon of brute force

These have been collected from books, other courses, and actual interview questions. Many of these questions can be easily solved with a hash table, but that requires extra linear storage. Try to solve without hash tables.

For any solution you come up, you must perform a running time analysis. Needless to say, try to find the solution with best possible running time.

Tips:

• In general, start with the brute force solution, analyze it, and try to see where improvements are possible.

• Given some problem on an unsorted array, ask yourself: does it becomes easier if you sort the array?

• Remember all your sorting related procedures: binary search, merging, pivoting. Some problems can be solved by adapting these procedures.

• One of the typical approaches is try to get an $O(n)$ factor into a $O(\log n)$ factor. Then (in some cases), you can bring the $O(\log n)$ down to $O(1)$.

• In many cases, divide and conquer is the way to go, *even though the problem might have no obvious way to split it.*

• Related to the previous points: in many cases, the brute force solution will involve some kind of "linear search" ($O(n)$ time search) over some quantity. Try to get this down to $O(\log n)$, with a clever search.

1. Given an integer x and an array of numbers, find if two numbers in the array sum up to x. (Or, find the pair of numbers whose sum is closest to x.)

2. Given two sorted arrays, find the median of the union of elements.

3. Given two sorted arrays, find the kth smallest element in the union of elements.

4. Do the above two questions when there are ℓ sorted arrays. What is the running time?

5. We discussed in class how to merge two sorted array. What about merging k sorted arrays?

6. You have an unsorted array of integers where every integer appears exactly twice, except one integer that appears once. Find it.

7. Given an array, describe an algorithm to identify the contiguous) subarray with the maximum sum. For example, if the input is $[1, -3, 5, -2, 9, -8, -6, 4]$, the output should be $[5, -2, 9]$. Get an algorithm that is $\Theta(n^2)$ (you can do much better than that).

8. Given an array A of numbers, for every index i, find the nearest index j such that $A[j] > A[i]$. If none exist, report -1. Output these indices sorted by the corresponding i.

9. Given an array of integers, move all the zeroes to the right end. The order of the others doesn't matter. Use $O(1)$ extra storage.

10. Given an array of integers, reorder so that all the negative integers appear first, then the zeroes, and finally the positive integers. If two integers have the same sign, their order should not be changed. Use $O(1)$ extra storage.

11. You have a stream of numbers. Implement an algorithm to return the median of the numbers seen so far.

12. A majority element in an array is an element that appears more than $n/2$ times (where n is the size of the array). Give an algorithm to find a majority element. More generally, give an algorithm to find all elements that appear more than $n/3$ times. Use $O(1)$ extra storage.

13. Given an array of integers containing all the elements from $1, 2, \ldots, n$ *except* one of them. Find which one. Try doing it using $O(1)$ memory. What if two elements were missing?

14. Given array of integers, determine if it contains a Pythagorean triple (integers a, b, c such that $c^2 = a^2 + b^2$).

15. The input is a sequence of ranges $[x_1, y_1]$, $[x_2, y_2]$, ... (where each $x_i \leq y_i$). Given two ranges that intersect, merge then into a single range by taking the union of ranges. Keep doing this until you finally get disjoint ranges. Design an algorithm that determines this output.

16. Given three arrays in non-decreasing order, find the elements that are common to all of them. Try $O(1)$ extra storage.

17. Consider two sorted arrays that are identical, *except* one array has an extra element. Find that element and its index.

18. Given a sorted array, find the closest element (in terms of absolute difference) to a given number.

19. Given an array, a local maximum is an element $A[i]$ such that is greater than or equal to the neighboring elements in the array. (There can be two neighboring elements, or just one if $i = 0$ or i is the last index.) Find a local maximum.

20. In a 2D array, a local maximum is defined as before, but the neighbors are both vertices and horizontal. So the neighbors of $A[i][j]$ are $A[i-1][j], A[i+1][j], A[i][j-1], A[i][j+1]$ (whenever they exist). Find a local maximum.

21. You are given only the head pointer of a singly linked list. Determine if the linked list has a loop. (It is not difficult with $O(n)$ storage, where n is the length of the list.) You *cannot* assume that the nodes store different values. The real challenge is to do this question with $O(1)$ extra memory.

22. Going beyond the problem above, find the first node on the loop. (The first node means that first node on the loop that is encountered, if you traversed from the head.) Use $O(1)$ extra memory.

23. Consider the "stock market" problem. There is an input array A of prices, of (say) one unit of stock. You need to find out the maximum profit that can be made, assuming you buy (one unit of) stock on a day and sell it on a subsequent day.

24. The stock market problem again. Except, you need to determine the best selling day, for every single day. That is, for each i, find $j > i$ that maximizes $A[j] - A[i]$.

25. Every array A induces a permutation π where $\pi(i)$ is the final location of the element $A[i]$ in the sorted version of A. Thus, element $A[i]$ ends up at index $\pi(i)$ after sorting. Given A, determine this permutation π.

26. Consider a version of MergeSort that does a 1/3-2/3 split, instead of in halves. Meaning, the "left" subarray will be the first third of elements, and the "right" array is formed by the remaining elements. What is the running time of this version of MergeSort?

27. Given a sorted array A of distinct integers, determine if there is an index i such that $A[i] = i$.

28. There is an extremely large array A such that the first n entries are positive integers, and all remaining entries are 0. Find n.

29. Consider an $n \times n$ 2D array where all rows and columns are in sorted order. Equivalently, the following holds for any i, j: for $i' > i$, then $A[i'][j] > A[i][j]$, and for $j' > j$, $A[i][j'] > A[i][j]$. Given an x, find if x is in the array. With some thought, a $\Theta(n \log n)$ algorithm is not hard. With more thought, you can improve on that.