CSCI2100: Quiz 2

Name: Student ID:

Problem 1 (20%). Suppose that we use quick sort to sort the array $A = (35, 12, 5, 55, 43, 78, 90, 82)$. Remember that the algorithm first randomly picks a pivot element from A and then solves two subproblems recursively. Let us assume that the pivot is 35. What are the input arrays of those two subproblems, respectively?

Solution. $(12, 5), (55, 43, 78, 90, 82)$.

Problem 2 (20%). Let A be the following array of 10 integers: (8, 5, 6, 2, 12, 1, 10, 17, 11, 9). Suppose that we use counting sort to sort the array, knowing that all the integers are in the domain from 1 to 20. Recall that the algorithm (as described in the class) generates an array B where each element is either 0 or 1. Give the content of B. You are reminded that in this course we adopt the convention that array indexes start from 1 (i.e., the first element of B is $B[1]$).

Solution. $(1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0)$.

Problem 3 (60%). Let S_1 be a set of n integers that have been sorted in an array. Let S_2 be another set of m integers that have *not* been sorted. Answer the following questions.

1. (30%) Give an algorithm to find $S_1 \cap S_2$ in $O(m \log n)$ time.
2. (30%) Suppose that all the integers in S_1 are in the domain from 1 to $100n$ (whereas the domain for S_2 is arbitrary). Give an algorithm to find $S_1 \cap S_2$ in $O(n + m)$ time.

Solution.

1. Let A_1 be the array storing S_1. For each integer $e \in S_2$, check whether $e \in S_1$ with binary search and, if so, output e. Each binary search costs $O(\log n)$ time. Thus, the total cost is $O(m \log n)$.
2. Let A_1 be the array storing S_1. Discard from S_2 all the integers that are outside the range $[1, 100n]$. Use counting sort to sort (the remaining elements of) S_2 in $O(m+100n) = O(m+n)$ time; let A_2 be the sorted array. Then, perform a synchronous scan over A_1 and A_2 to output $S_1 \cap S_2$ as follows. First, set $i = 1$ and $j = 1$. Then, repeat the following until $i > |A_1|$ or $j > |A_2|$: if $A_1[i] = A_2[j]$, output $A_1[i]$ and increase both i and j by one. If $A_1[i] > A_2[j]$, increase j by one; if $A_1[i] < A_2[j]$, increase i by one. The synchronous scan takes $O(m + n)$. So the overall cost is $O(n + m)$.