EECS 336 Fall 2015
Homework Problem 4.3

Let $A = \{a_1, a_2, ..., a_l\}$ with $l \leq n$, $B = \{b_1, b_2, ..., b_k\}$ with $k \leq n$ with elements in both arrays as integers no larger than n. Assume both A and B are sorted in ascending order ($a_i < a_{i+1}$ for $i \leq l$ and $b_j < b_{j+1}$ for $j \leq k$), which requires $O(n \log(n))$ time.

a: Brute-force approach. Construct C as a dictionary with key as values of the sums of elements from A, B and key as counts of each sum.

Algorithm 1 Brute-force calculation of sums

1: **procedure** BFSums(A, B)
2:     Initialize C as an empty dict.
3:     **for** each element a_i in A **do**
4:         **for** each element b_j in B **do**
5:             $c_p = a_i + b_j$
6:             **if** C has key c_p **then**
7:                 Update $C[c_p] \leftarrow C[c_p] + 1$
8:             **else**
9:                 Create key $C[c_p] \leftarrow 1$
10:             **end if**
11:         **end for**
12:     **end for**
13: **end procedure**

Correctness: By contradiction. Suppose the algorithm produces neither the correct sum of elements from A, B nor correct of count for each sum. Then according to Line 5, for each pair of elements a_i, b_j, sum $c_p = a_i + b_j$ is included in C as a key, which rather provides a correct solution. Moreover, from the algorithm, value of key $C[c_p]$ corresponds to the times of appearances of c_p as a result of addition of elements a_i, b_j. Therefore, both values and counts of sums are correctly calculated in dictionary C.

Runtime Analysis: Both for loops iterate n times since size of A, B are of $O(n)$, with constant time during each iteration for operations of additions and dictionary update. Thus, constructing C takes $O(n^2)$. Since initial sorting of A, B takes $O(n \log(n))$, total runtime with brute-force approach is $O(n^2)$.

b: Let A', B' be polynomials with respect to x up to powers of a_l, b_k, respectively:

$$A' = x^{a_1} + x^{a_2} + ... + x^{a_l}$$
$$B' = x^{b_1} + x^{b_2} + ... + x^{b_k}$$

Here, a_l, b_k are the same elements as in A, B. Multiply polynomials A', B' using iterative FFT as covered in class to derive $C' = A'B'$:

$$C' = h_1 x^{a_1+b_1} + ... + h_p x^{a_l+b_k} = h_1 x^{c_1} + h_2 x^{c_2} + ... + h_m x^{c_m}$$

Here, powers of terms in polynomial C', c_p with $p \leq l+k$, constitute the same set as required:

$$C = \{a + b \,|a \in A \wedge b \in B\}$$

with coefficients of each term h_q corresponding to the counts that each term appears as a result of multiplication of elements in A, B.

Correctness: By contradiction. Suppose algorithm outputs C' such that either (1) not all sums of $c_k = a_i + b_j, \forall a_i \in A, b_j \in B$ are expressed as powers in some term in C', or (2) the coefficients c_k in C' incorrectly shows the sum of $a_i \in A, b_j \in B$ such that $c_k = a_i + b_j$.

For (1), from the algorithm, since the multiplication of polynomials produces $C' = A'B'$, thus for some $x^{a_i} \in A', x^{b_j} \in B'$, there exists a term $x^{a_i+b_j} \in C'$ that accounts for the sum $c_k = a_i + b_j$.

For (2), for each pair of $a_i \in A, b_j \in B$ that satisfy $c_k = a_i + b_j$, one term of x^{c_k} will be added to C', which increments the coefficient h_k of this term by one. Then, count of appearance of $x^{c_k} \in C'$ is exactly equal to number of pairs of $a_i \in A, b_j \in B$ that satisfy $c_k = a_i + b_j$.

Thus, the algorithm produces C' as the correct result of polynomial with power of each term as the sum and coefficient of each term as the count of the sum.

Runtime Analysis: Part b gives a formulation of the original problem as multiplicaton of two polynomials, which can then be solved with iterative FFT in $O(n \log(n))$. Since initial sorting of A, B takes $O(n \log(n))$, total runtime with brute-force approach is $O(n \log(n))$.