Proof of Correctness

Since Algorithm 2 is essentially the same as Algorithm 3 but operates under an assumption on the input, let us concentrate on Algorithm 3 only. Also, we do not need to stop Algorithm 3 after it prints k sums. Let us allow it to print all the AB-sums (that is, run the loop until Q becomes empty).

Claim: *Algorithm 3 prints the AB-sums in ascending (actually, non-decreasing) order.*

Proof Suppose that at some point of time, s is the smallest among the sums yet to be printed. It is possible that a few instances of s have already been printed. But what is important is that at least one more instance of s remains to be printed. We prove that there exists a pair (i^*, j^*) such that $A[i^*] + B[j^*] = s$, and $Q[1] = (i^*, j^*)$.

We first show that there is an (i, j) such that $A[i] + B[j] = s$, and (i, j) was inserted in Q. Let

$$X = \{(i, j) \mid A[i] + B[j] = s, \text{ and } (i, j) \text{ has not been printed so far}\}.$$

X is non-empty and so must have a minimal element (i, j). Here, the minimality of (i, j) means that for no $(i', j') \in X$, $(i', j') \neq (i, j)$, we have both $i' \leq i$ and $j' \leq j$ with at least one inequality strict. If $(i, j) = (1, 1)$, then it has already been inserted in Q in the initialization step.

So suppose that $(i, j) \neq (1, 1)$, that is, either $i \geq 2$ or $j \geq 2$. Take the case $i \geq 2$ (the other case is analogous). We have $\lfloor i/2 \rfloor \geq 1$, that is, $(\lfloor i/2 \rfloor, j)$ is a valid index pair. Also, the heap ordering of A implies that $A[\lfloor i/2 \rfloor] + B[j] \leq A[i] + B[j]$. If $A[\lfloor i/2 \rfloor] + B[j] < A[i] + B[j]$, then $(\lfloor i/2 \rfloor, j) \notin X$ by the definitions of s and X, whereas if $A[\lfloor i/2 \rfloor] + B[j] = A[i] + B[j]$, then $(\lfloor i/2 \rfloor, j) \notin X$ by the minimality of (i, j) in X. In either case, we see that $(\lfloor i/2 \rfloor, j)$ has been printed. But then (i, j) must have been inserted in Q.

Now, note that $A[i] + B[j]$ corresponds to the minimum AB-sum currently stored in Q, because Q contains a subset of the index pairs yet to be printed. Therefore if $Q[1] = (i^*, j^*)$, we must have $A[i^*] + B[j^*] = A[i] + B[j] = s$. We may have $(i^*, j^*) = (i, j)$ but this is not necessary. •

Algorithm 4

(Proposed by Anurag Anand)

1. Convert A and B individually to two min-heaps.
2. Store the k smallest elements of A in an array $a[1 \ldots k]$, and the k smallest elements of B in an array $b[1 \ldots k]$. This can be done by *deletemin*'s from A and B. If you want to keep the compositions of A and B, store the deleted minimums at the empty cells created by the deletions (as in heap sort).
3. Initialize a priority queue Q by $(i, 1)$ for $i = 1, 2, \ldots, k$. Q is ordered with respect to $a[i] + b[j]$.
4. Repeat k times:

    Let (i, j) be obtained by *extractmin* from Q. Print $a[i] + b[j]$. If $j < k$ (or if $i + j \leq k$), insert $(i, j + 1)$ in Q.

Exercise: Formally establish the correctness of Algorithm 4.