Find K Pairs with Smallest Sums

Goal:

Produce the k pairs with the smallest sums from two sorted (non-decreasing) arrays.

Key Observations:

Because both arrays are sorted, for any fixed i, the pairs (nums1[i], nums2[0]), (nums1[i], nums2[1]), ... have non-decreasing sums. So for each i, the "best candidate" to try first is (i, 0). If you ever take (i, j), the next candidate from that row is (i, j+1).

Greedy + heap strategy (k-way merge style):

1. Initialize a mean-heap keyed by nums1[i] + nums2[j]. Seed it with the first pair from up to min(k, n1) rows (i, 0) for i = 0... min(k, n1)−1. (No need to seed more than k rows because you'll extract at most k pairs).
2. Repeat up to k times (and stop early if the heap empties):
   - Pop the smallest-sum pair (i, j) -> output (nums1[i], nums2[j]).
   - Push the next pair from the same row if it exists: (i, j+1).

This works because you always expand the smallest unseen candidate next, exactly like merging k sorted lists.

Correctness sketch:

- The heap always contains the frontier of the smallest not-yet-output pairs (one per active row).
- When you pop (i, j), any pair smaller than it must have already been inserted earlier; any larger candidate from row i is (i, j+1), which you now add.
- By induction, outputs are in non-decreasing sum order, yielding the first k.

Complexity:

- Heap holds at most $\min(k, n_1)$ elements.
- Each of up to k pops can trigger at most one push
- Time: $O(k \log \min(k, n_1))$.
- Space: $O(\min(k, n_1))$