1. Greedy, similar to median selection. $O(n+k)$.
2. Counting sort. $O(n+U)$.
3. Use the result for selecting the k-th largest element in the union of n sorted arrays. $O(n \log \frac{k}{n})$ (for $k \geq n$). See 004. Median of Two Sorted Arrays.
4. Group the numbers into buckets with range $[2^i, 2^{i+1})$. An element in the i-th bucket will go to the $(i-1)$-th bucket after an operation. First use weighted median selection to find the bucket where the k-th largest element belong to in $O(\log U)$ time. Then perform median selection within the bucket. $O(n)$ (we can eliminate the $O(\log U)$ term using bit operations).

Remove Stones to Minimize the Total

Submission Detail

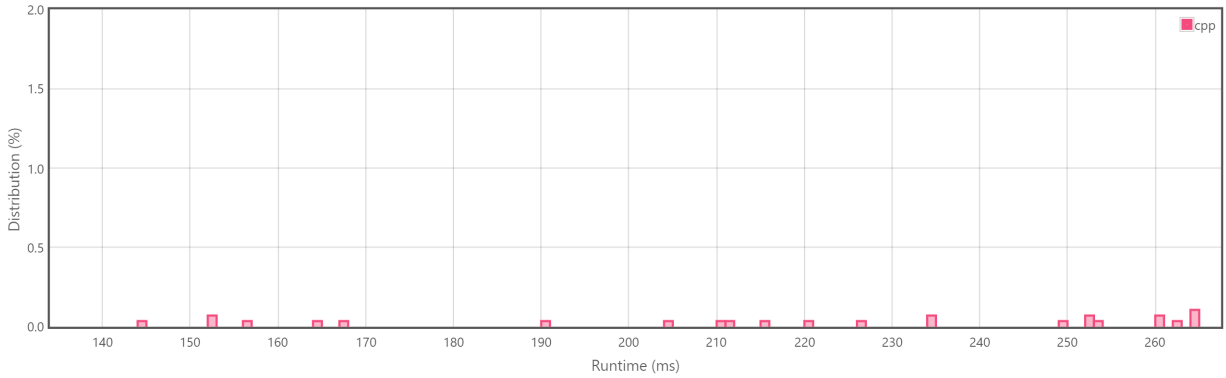
59 / 59 test cases passed.

Status: Accepted

Runtime: **64 ms**
Memory Usage: **94.3 MB**

Submitted: **0 minutes ago**

Accepted Solutions Runtime Distribution



Runtime: 64 ms, faster than 100.00% of C++ online submissions for Remove Stones to Minimize the Total.

Memory Usage: 94.3 MB, less than 100.00% of C++ online submissions for Remove Stones to Minimize the Total.

References