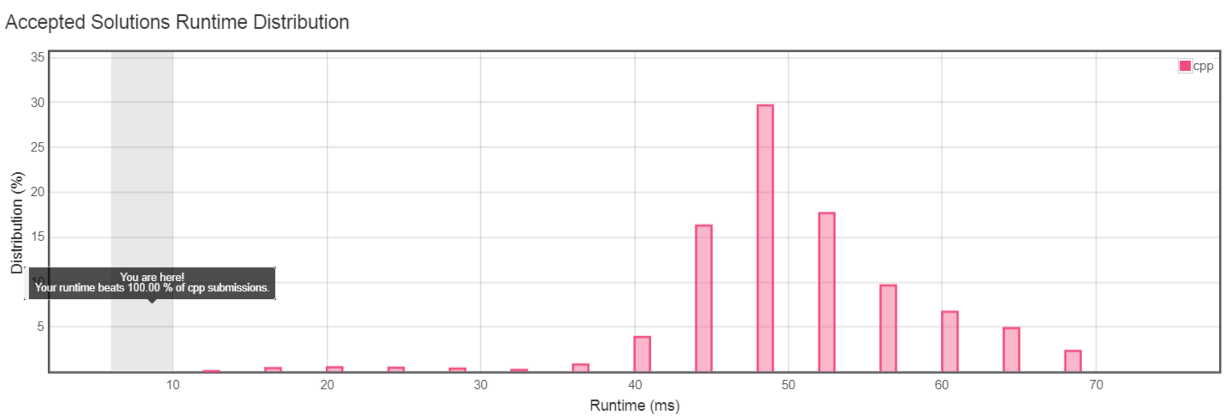
1. binary search. $O(n \log W)$.
2. Let d denote the divisor, we know $f(d) \triangleq \sum_{i=1}^{n} \lceil \frac{a_i}{d} \rceil \geq \sum_{i=1}^{n} \frac{a_i}{d}$. each $\lceil \cdot \rceil$ has additive error at most 1, so if we set $d = \frac{\sum_{i=1}^{n} a_i}{t}$, $f(d)$ is an approximation for t with additive error $O(n)$. if we maintain for each a_i the next value d s.t. $\lceil \frac{a_i}{d} \rceil$ will change (which can be computed in $O(1)$), then we need to test only $O(n)$ subsequent d's to make sure $f(d) \leq t$. this can be maintained by heap in $O(n \log n)$ time.
For a fixed d, we can estimate $f(d) - \sum_{i=1}^{n} \frac{a_i}{d}$ by sampling $O(n^c)$ elements to get an $\tilde{O}(n^{1-\frac{c}{2}})$ additive approximation, by Chernoff bound (since $0 \leq \frac{a_i \bmod d}{d} < 1$). Set $c = \frac{1}{2}$ suffices. Then first perform $O(\log W)$ binary search steps (each step in sublinear time), then use heap to test a sublinear number of subsequent d's (notice that build a heap only takes $O(n)$ time). $O(n + \log W \cdot n^{\epsilon})$, which is $O(n)$ if we reduce W to poly(n).

remark.
1. it should possible to perform only $O(\log n)$ binary search steps, by known techniques.
2. it seems we can use the results for finding the m-th largest element in the union of n sorted array [1, 2], and also get $O(n)$ time. see 004. Median of Two Sorted Arrays.



References

[1] Greg N Frederickson and Donald B Johnson. Generalized selection and ranking: sorted matrices. *SIAM Journal on computing*, 13(1):14–30, 1984.

[2] Andranik Mirzaian and Eshrat Arjomandi. Selection in x+ y and matrices with sorted rows and columns. *Information processing letters*, 20(1):13–17, 1985.