

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  $c = \frac{2}{3}$  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).  $O(n + \log W \cdot n^{\frac{2}{3}})$ , which is  $O(n)$  if we reduce  $W$  to poly( $n$ ) by standard techniques.

3. We can directly use the results for finding the  $m$ -th largest element in the union of  $n$  sorted arrays [2, 3, 4], and also get  $O(n + \min\{n, m\} \log \frac{m}{\min\{n, m\}}) = O(n)$  time (since  $m = O(n)$ ). see 004. Median of Two Sorted Arrays.

There's a simpler algorithm that finds the  $m$ -th largest element in the union of  $n$  sorted array in  $O(n + m)$  time for this special case, using linear-time median selection. Each round using  $O(n)$  time, we either reduce  $n$  by half, or decrease  $m$  by at least  $\frac{n}{2}$ .

4. Store the numerators in  $O(U)$  bins. When we binary search for the divisor  $d$ , we can compute the sum in  $O(\frac{U}{d})$  time, using prefix sum.  $O(\sum_{0 \leq i \leq \log U} \frac{U}{2^i}) = O(U)$ .

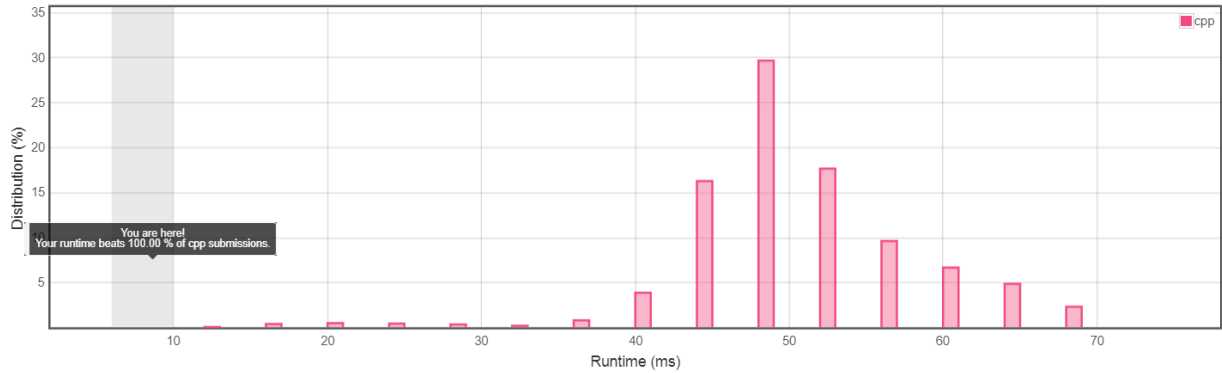
Remark.

1. it's possible to perform only  $O(\log n)$  binary search steps, by known techniques. (extract the first  $n$  largest elements in the union of  $n$  sorted arrays, after using the observation of Alg. 2, and then binary search on those  $n$  values.)

2. with clever implementation we don't need long long.

3. a related paper: [1].

Accepted Solutions Runtime Distribution



## References

- [1] Zhanpeng Cheng and David Eppstein. Linear-time algorithms for proportional apportionment. In *International Symposium on Algorithms and Computation*, pages 581–592. Springer, 2014.
- [2] Greg N Frederickson and Donald B Johnson. Generalized selection and ranking (preliminary version). In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 420–428, 1980.
- [3] Greg N Frederickson and Donald B Johnson. The complexity of selection and ranking in  $X + Y$  and matrices with sorted columns. *Journal of Computer and System Sciences*, 24(2):197–208, 1982.

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