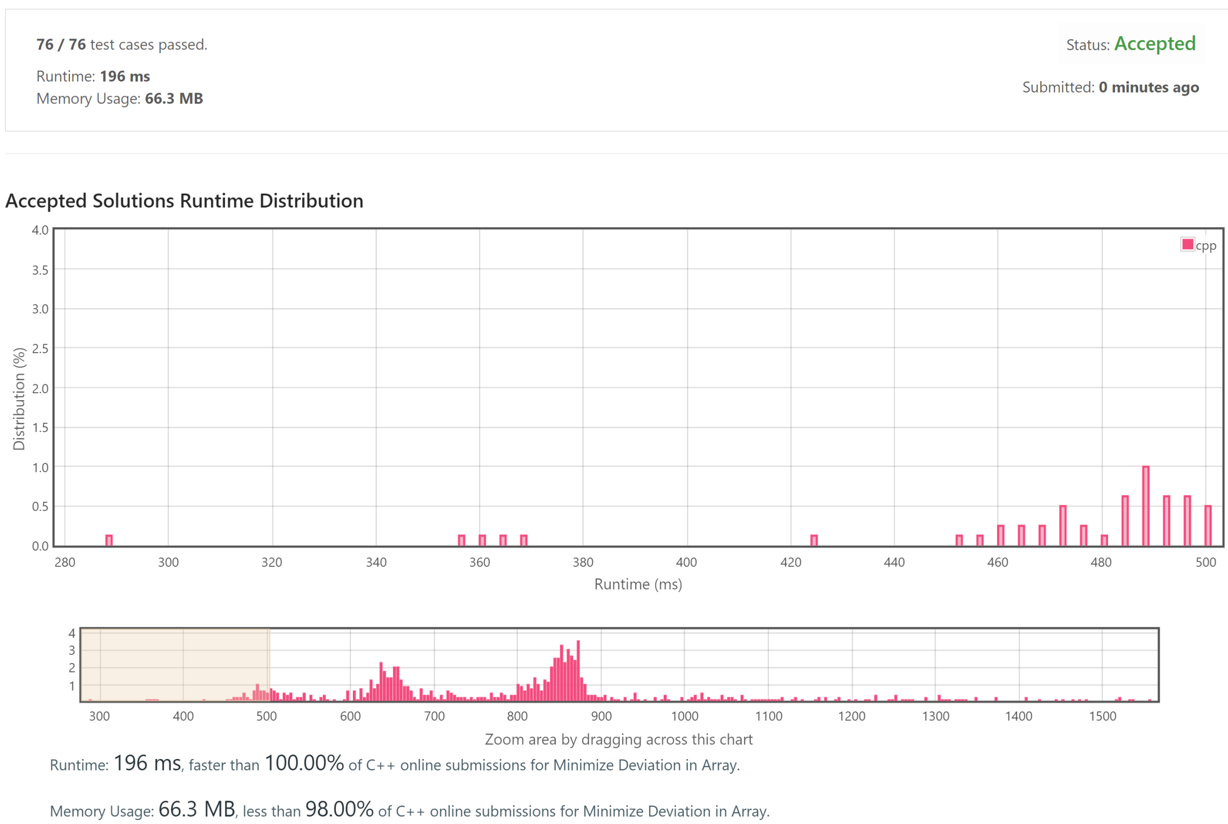
First repeatedly divide each number by 2 until it becomes odd. Let ma denote the largest value among them, we claim the optimal solution must include ma: otherwise suppose the optimal solution includes $ma \cdot 2^j$ ($j \geq 1$) instead, then any other number must be even, otherwise it's odd and $\leq ma$, the deviation $\geq ma \cdot 2^j - ma \geq ma$, which is not optimal. Then tune each number to the maximum possible number that $\leq ma$, denote as $a[i]$. Let $mi = \min_i a[i]$. The optimal solution must either use $a[i]$ or use $2 \cdot a[i]$ (if it's valid). In other words, the optimal solution must be of the following form: choose an index i, for each $a[j] \leq a[i]$ use $2 \cdot a[j]$ (and should be valid), otherwise use $a[j]$. Let $\mathrm{succ}(x)$ denote the successor of x, then the deviation is $2 \cdot x - \mathrm{succ}(x)$. If multiplying mi by 2 is valid, then we have $2 \cdot mi > ma$, i.e. $mi > \frac{ma}{2}$; otherwise the optimal solution is $ma - mi$.

In order to avoid sorting, the next key observation is if there are multiple values in $[mi+2^j, mi+2^{j+1})$, then only selecting the largest value among them may be optimal: for any other number $x \in [mi+2^j, mi+2^{j+1})$, $2 \cdot x - \mathrm{succ}(x) > 2 \cdot x - (mi + 2^{j+1}) \geq mi$, but the optimal solution is at most $ma - mi < mi$. Thus we can use $O(\log W)$ bins, and record the minimum and maximum value in each bin. To compute the optimal solution, enumerate the bins in increasing order, and we can terminate early if we find a bin containing any number that cannot be multiplied by 2. $O(n)$.



References