assume an ugly number can be stored in a word.
1. heap+hashing. $O(n \log n)$.
2. for each prime 2/3/5, maintain a pointer pointing to the ugly number which can multiply with the prime to generate the next ugly number. $O(n)$.
3. there are $O(\log^3 x)$ ugly numbers that $\leq x$, and here $n = O(\log^3 x)$. implicitly construct an $O(\log x) \times O(\log x) \times O(\log x)$ 3D sorted matrix, where entry (i, j, k) denote $2^i \cdot 3^j \cdot 5^k$. we can find the n-th ugly number in $O(n^{\frac{2}{3}} \log n)$, by finding the n-th smallest element in $O(\log x) \times O(\log x)$ sorted lists. (we can possibly remove the $\log n$ factor)
see 378. Kth Smallest Element in a Sorted Matrix and 4. Median of Two Sorted Arrays.

note. consider the following problem, which is useful to the original problem: given value x, compute the number of pairs (i, j) s.t. $2^i 3^j \leq x$.
the naive complexity is $O(\log x)$ by enumerating $i = O(\log x)$ and use monotone pointer to maintain j.
can we use gcd? after taking log, reduce to computing the number of grid cells on a plane under line $ax + b$. However, a and b need precision $O(\log x)$ bits, so gcd takes $O(\log x)$.
Now we lower the precision to get $a_1 x + b_1$ and $a_2 x + b_2$ as the lower bound and upper bound for $ax + b$, using t bits. gcd takes $O(t)$. We can find all grid points between $a_1 x + b_1$ and $a_2 x + b_2$ by binary search (and verify by gcd, according to whether the results for $a_1 x + b_1$ and $a_2 x + b_2$ differ), assume there are k such points, this takes $kt \log \log x$, then use $O(k)$ to check each of them. If we assume $i \log 2 + j \log 3$ has some good pseudo-randomness property, $k = O(\log x) \cdot e^{-O(t)}$. choose $t = O(\log \log x)$, the total running time is $O(poly \ \log \log x)$.
金斌：欧几里得算法的应用

References