k-Smooth Numbers and its Generalization

Zhenhua Lin

May 20, 2011

Keywords: smooth number, Hamming number, Hamming problem, regular number, algorithm

1 Introduction

A positive integer is called k-smooth if none of its prime factors are greater than k. Let $P(k)$ be the set of prime numbers not greater than k, i.e.,

$$P(k) = \{n \in \mathbb{N} : n \text{ is prime number and } n \leq k\}.$$

In the classic Hamming problem, we are asked to print the first n 5-smooth numbers (they are also called Hamming numbers) in the increasing order. Dijkstra [1] proposed the algorithm below in 1981 to solve this problem.

Algorithm 1 Dijkstra's 5-smooth algorithm

```
  $H \leftarrow \{1\}$ , $k \leftarrow 0$, $R \leftarrow \emptyset$
  while $k < n$ do
    $h \leftarrow$ the minimum of $H$.
    $H \leftarrow 2H \cup 3H \cup 5H$ ($aH$ means the set $\{ah : a \in H\}$ )
    Put $h$ into $R$
    $k \leftarrow k + 1$
  end while
```

Although Dijkstra's algorithm is designed to solve the classic Hamming problem, it is quite straightforward to extend it to solve general Hamming problem, i.e., print the first n k-smooth number in the increasing order. However, the programming language without lazy evaluation feature, it is very hard to manage it to run in $O(n)$ time, provided that computing multiplication of two integers takes $O(1)$ time. In this paper, we propose a new algorithm to solve Hamming problem in a more general setting described below.

Let B be a finite set of m positive integers. We arrange the elements in B so that $b_1 < b_2 < \ldots < b_m$. We say that B is a smooth base if for each element b_k, it has a prime factor p_k whose is not included in any other element. Formally, B is a smooth base if and only if

$$\forall k \, \exists p_k : \, (p_k \mid b_k) \wedge (\forall q \neq k : \, p \nmid b_q),$$

where $p \mid b_k$ means there is a integer t such that $b_k = pt$, and $p \nmid b_k$ means no such integer t exsits. Such a prime factor p_k is called the key factor of b_k. In other words, B is a smooth base if and only if every element has at least one key factor.

Given a smooth base $B = \{b_1, b_2, \ldots, b_m\}$, we use $H(B)$ to denote the set containing all the integers in the form

$$b_1^{x_1} b_2^{x_2} \cdots b_m^{x_m},$$

where $x_1, x_2, \ldots, x_m$ are nonnegative integers. We also say that $H(B)$ is generated by B, and numbers in B are called generalized Hamming numbers. It is easy to see that every number in $H(B)$ can be uniquely represented by this way, i.e., every number in $H(B)$ correpsonds to a unique tupple $(x_1, x_2, \ldots, x_m)$. Thus, once the smooth base is fixed, we also simply use $(x_1, x_2, \ldots, x_m)$ to denote $b_1^{x_1} b_2^{x_2} \cdots b_m^{x_m}$.

Note that $P(k)$ is a smooth base for all $k \geq 2$. For example, if $B = P(5) = \{2, 3, 5\}$, then $H(B)$ is the collection of all 5-smooth numbers. Thus, computing the first n k-smooth numbers is just a special case of computing first n generalized Hamming numbers generated by a smooth given smooth base.

2 Algorithm

If examinating the Dijkstra's algorithm closely, we can see that there are two essential operations: finding the minimum of H and merging $2H$, $3H$ and $5H$. For the merging operation, if the three components merged are disjoint, then it is very easy. Unfortunately, they are not. For example, 2×3 belong to $2H$ and $3H$ for $H = \{2, 3, 5\}$. To avoid such duplicating, one possible way is to resolve the ambiguity: assign 2×3 to either $2H$ or $3H$, but not both. This can be done by adopting a kind of "Maximum Principle": if $a \in sH$ and $a \in tH$ at the same time, and $s < t$, we only assign a to tH. For a belongs to more than two such sets iH, we assign a to the one with largest i. Inspired by this idea, we have following algorithm to compute first n generalized Hamming numbers generated by a smooth base $B = \{b_1, b_2, \ldots, b_m\}$.

Theorem 1. *When Aglorithm 2 terminates, R is the sequence of the first n generalized Hamming numbers generated by B.*

To prove the theorem, we shall firstly establish two impartant facts. The first one shows that Q_i and $Q_j$are disjoint if $i \neq j$.

Fact 1. *If $i \neq j$, then $Q_i \cap Q_j = \emptyset$.*

Proof. We first show by induction that any element in Q_1 does not contain any key factors of $b_2, b_3, \ldots, b_m$. At the beginning, $Q_1 = \{b_1\}$ and by the definition of smooth base B, b_1 does not have key factors of $b_2, b_3, \ldots, b_m$. Now assume after the first p loops of the while block, elements in Q_1 do not contain key factors of $b_2, b_3, \ldots, b_m$. If at the $p + 1$st loop of the while block, a new element

Algorithm 2 Algorithm for Computing $H(B)$

```
Initialize queues $Q_1, Q_2, \ldots, Q_m$ to be empty
$R \leftarrow \emptyset$
for $t$ from 1 to $m$ do
    push $b_t$ into queue $Q_t$
end for
$k \leftarrow 0$
while $k < n$ do
    Let $h$ be the minimum element in the front of each queue $Q_i\,(1 \leq i \leq m)$
    and assume $h \in Q_j$
    for $t$ from $j$ to $m$ do
        push $h \cdot b_t$ into queue $Q_t$
    end for
    Remove $h$ from $Q_j$.
    Put $h$ into output sequence $R$
    $k \leftarrow k + 1$
end while
```

$h \cdot b_1$is pushed into Q_1, then according to the algorithm, h must come from Q_1 and hence h does not contain key factors of $b_2, b_3, \ldots, b_m$. This also implies that $h \cdot b_1$ does not contain key factors of $b_2, b_3, \ldots, b_m$ and all members of Q_1 do not contain key factors of $b_2, b_3, \ldots, b_m$ in the end of the $p+1$st loop. The statement is trivially true if there is no element pushed into Q_1in $p+1$st loop.

Again, by induction we show that members in Q_i $(1 \leq i < m)$ do not contain key factors in $b_{i+1}, b_{i+2}, \ldots, b_m$. For $i = 1$, the statement holds by the argument above. Now assume the statement holds for all $1 \leq i < p < m$. By the similar argument used to prove the statement for Q_1, we can show that the statement also holds for Q_p. Therefore, the statement holds for $1 \leq i < m$.

By the algorithm, we also know that elements in Q_i $(1 \leq i \leq m)$ includes key factor of b_i. Now let $1 \leq i < j \leq m$. Since members in Q_i do not contain any key factor of b_j while all members in Q_j have a key factor of b_j, no element in Q_i can belong to Q_j and hence $Q_i \cap Q_j = \emptyset$. $\square$

Now we have known that elements from different queues are distinct. To demonstrate that no duplicated numbers will be added to R, we need to show that elements in the same queue is also distinct. Actually, we manage to show a stronger conclusion: elements in the same queue are pushed into the queue by the strictly increasing order. We also obtain an important fact at the same time, which shows every generalized Hamming number will be pushed into some queue. This gurantees than no generalized Hamming numbers are skipped by the algorithm.

Before starting the next fact, we define followers of a generalized Hamming number $(x_1, x_2, \ldots, x_m)$ as the m numbers $(x_1 + 1, x_2, \ldots, x_m)$, $(x_1, x_2 + 1, \ldots, x_m), \ldots, (x_1, x_2, \ldots, x_m + 1)$.

Fact 2. *At any time, elements in Q_i $(1 \leq i \leq m)$ are strictly increasing and* *hence distinct. Also, if at step p, $(x_1, x_2, \ldots, x_m)$ is removed from some queue, then each of its m followers, either has been already pushed into some queue at some step q ($q < p$), or will be pushed into some queue at the step p.*

Proof. Again, we prove it by induction. Obviously, at the very beginning of the first loop of the while block, the statement above holds. Assume the statement is correct at the step $p-1$. Suppose at the step p, $g = (x_1, x_2, \ldots, x_m)$ is removed from queue Q_j.

For $1 \le i < j$, since $(x_1, \ldots, x_i + 1, x_{i+1}, \ldots, x_j - 1, x_{j+1}, \ldots x_m)$ is smaller than g, it was removed at some step $l < p$, according to the induction assumption. Therefore, by the assumption, its follower $(x_1, \ldots, x_i + 1, x_{i+1}, \ldots, x_m)$, which is also a follower of g, was pushed into some queue before the step p. For $i \ge j$, the follower $(x_1, \ldots, x_i + 1, \ldots, x_m)$ is pushed into Q_i at the step p. Therefore, the second half of the statement still holds for the step p.

For each i such that $j \le i \le m$, there is a new element $(x_1, \ldots, x_i+1, \ldots, x_m)$ pushed into Q_i. If Q_i contains only one element $(x_1, \ldots, x_i + 1, \ldots, x_m)$, then Q_i is increasing trivially. Now assume Q_i contains more than one element. Let $(y_1, \ldots, y_i + 1, \ldots, y_m)$ be any one in Q_i other than $(x_1, \ldots, x_i + 1, \ldots, x_m)$. According to the algorithm, $(y_1, \ldots, y_i, \ldots, y_m)$ was pushed into Q_i before g because the element $(y_1, \ldots, y_i-1, \ldots, y_m)$ was removed from some queue before g. Hence $(y_1, \ldots, y_i - 1, \ldots, y_m) < g$, and further $(y_1, \ldots, y_i - 1, \ldots, y_m) \times b_i < g \times b_i$. That is, $(y_1, \ldots, y_i, \ldots, y_m) < (x_1, \ldots, x_i + 1, \ldots, x_m)$, and Q_i is strictly increasing. □

Given these two facts established, it is quite straightforward to see the correctness of the statement in Theorem 1.

Proof. Since numbers in each queue is strictly increasing and numbers in all queues are distinct, the outputed sequence is strictly increasing and hence has no duplicates. Also, since no number will be skipped, the output sequence must contain the first n generalized Hamming numbers generated by the base B when the algorithm terminates. □

References

[1] Edsger W. Dijkstra. Hamming's exercise in sasl. 1981.