Compute the prefix-sum by powerful number sieving

baihacker
2020.04.07

Method description

Consider computing the prefix-sum of a multiplicative function f, the prefix-sum is given by **cal**(index, n). Index is the smallest number where the prime[index] * prime[index] > n. (index starts from 0)

```
int64 cal(int64 i, int64 n) {
  int64 ret = sg(n);
  for (int j = 0; j < i; ++j) {
    int64 p = plist[j], m = n / p / p;
    if (m == 0) break;
    for (int k = 2; m >= 1; ++k, m /= p) if (h(k) != 0) {
      ret += h(k) * cal(j, m);
    }
  }
  return ret;
}
```

- The definition of f and g will be discussed in a later section.
- sg is the prefix-sum of g.
- plist[i] is the i_th prime (index starts from 0).

This method can be viewed as $\sum_{p \text{ is powerful or } 1} h(p) sg(\frac{n}{p})$ and that's why we call it powerful number sieving.

Find f and g

To find multiplicative function h, g, consider their definitions on p^k (p is prime) we have $f(p^k) = \sum_{i=0}^{k} h(p^i) g(p^{k-i})$.
Then, we have the way to:

- Find g

    - We can start from $f(p) = g(p)$.
- Find h
    - In the method, we skip p^2, to make sure it is correct, we add a forced constraints $h(p) = 0$.
    - For $k \geq 2$, we can get the value of $h(p^k)$ by solving an equation since g is found and $h(p^i)$ is known for $i < k$.

Complexity

By the view of powerful number sieving, since a powerful number can be represented as a^3b^2, it is easy to see the complexity is $\iint_{x,y} O_{sg}(\frac{n}{x^3y^2})dx\,dy$, so the complexity is

$$\begin{cases} n^{1/2} & O_{sg} \text{ is smaller than } n^{1/2} \\ n^{1/2}\log n & O_{sg} \text{ is } n^{1/2} \\ O_{sg} & O_{sg} \text{ is larger than } n^{1/2} \end{cases}$$

So, if we want to make use of this method, another constraint on g is that we can compute sg fast.

Note: we assume that $h(p^k)$ can be computed in a reasonable complexity.

Further thoughts

Reach n^(⅓)

If we have $h(p) = h(p^2) = 0$, we may have an algorithm of complexity around $n^{\frac{1}{3}}$. But meanwhile, another constraint is added on g, $f(p^2) = g(p^2)$. The challenges are

- Can we find it?
- Can we compute sg in a reasonable complexity.
- Since $f(p^k) = g(p^k)$ for $k <= 2$, they are too similar. We want to reduce the complexity of computing sf but g is similar to f, so can we reduce a lot if they are too similar?

Similarity rank and complexities

We can define the similarity rank of two multiplicative functions by the maximum k such that $f(p^i) = g(p^i)$ if $i \leq k$. The question is, how does the similarity rank affect the prefix-sum computation complexity of f and g? More concrete: if k is given, what's the maximum complexity we can reduce. If we only consider polynomial complexity, what the maximum value of $\{cf - cg | \text{complexity of } f = O(n^{cf}), \text{complexity of } g = O(n^{cg})\}$ if f is given and the similarity rank of f and g is k.

If h(p) != 0

$\sum_{p \text{ is powerful or } 1} h(p)sg(\frac{n}{p})$ becomes $\sum_p h(p)sg(\frac{n}{p})$, we only need the value of $sg(i), sh(i), sg(\frac{n}{i}), sh(\frac{n}{i}), i \le n^{1/2}$. Let $O(n^{\frac{a}{b}}) = \max(O(sh), O(sg))$ (usually, we have $0 \le a < b$). So, based on $\int_1^n x^{\frac{a}{b}} + (\frac{n}{x})^{\frac{a}{b}} dx$, the complexity is $O(n^{\frac{a+b}{2b}})$. For example $a = 1, b = 2$, the complexity is $O(n^{\frac{3}{4}})$. When $a = 0$, the lower bound is $\Omega(n^{\frac{1}{2}})$. This lower bound is consistent with our intuition, i.e. we need to iterate $O(n^{\frac{1}{2}})$ function values.
This analysis also works for $h(p) = 0$, in which we only iterate on powerful numbers, so the complexity or the complexity constant will be smaller.
Compared to the analysis of powerful number sieving, $h(p) \neq 0$ requires that both sg, sh have a good complexity while the powerful number sieving depends on the value of sg on $O(n^{\frac{1}{2}})$ inputs and the value of h on $O(n^{\frac{1}{2}})$ inputs. Since h is considered to have a better complexity than sh, the powerful number sieving has a better performance. Another view is that we shift the complexity of sh to h's definition on square free numbers (excluding 1), i.e. 0.

More general approach

As mentioned by fjzzq2002 [1], $f(p^k) = \sum_{i=0}^k h(p^i)g(p^{k-i})$ can be viewed as the Dirichlet convolution of g and h. So, we can have a general approach to reduce complexity of compute the prefix-sum of a multiplicative function to represent it as the Dirichlet convolution of two other functions and consider how to compute based on the convolution format.
We have already had an example which is aligned to this approach: mobius inversion. I have another article [5] written in chinese which tries to generalize mobius inversion. It also considers finding the convolution representation, and discussing some ideas/guidance about how to use the convolution to reduce the complexity.
Overall speaking, the method in this article and mobius inversion are unified in this sense.

References

[1] fjzzq2002, 2018.11.01,Use powerful numbers to compute the prefix sum of multiplicative functions (chinese content)
[2] Min_25, 2018.02.04, solution for Counting modulo pairs (problem authored by baihacker, Min_25's solution link is not provided intentionally)
[3] abcwuhang, 2018.10.24, posts on Summing a multiplicative function (problem authored by abcwuhang)
[4] asaelr, fakesson, 2020.03.28, posts on Twos are all you need (problem authored by abcwuhang)

[5] baihacker, 2018.03.18, Thinking on the generalized mobius inversion (chinese content)