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)
      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

If we have $h(p) = h(p^2) = 0$, we may have an algorithm of complexity n. 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.
  - 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 of f and g is k.

References

[1] fjzzq, 2018.11.01, using powerful numbers to compute the prefix sum of multiplicative functions [link].

[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)