Tashfeen, Ahmad
Doctoral Research
16 June 2022

Tashfeen's Research Notes

Question 1. What is an efficient way to store prime numbers?

1.1. 32-bit Prime Numbers

The largest prime that can be saved using 32 bits is,

$$p_i = 4294967291 \quad \text{where} \quad i = 203280221$$

Note that $32 - \lg(p_i) < 0.000000002$. Therefore, taking 32 bits to save each of the $\pi(2^{32})$[1] prime numbers, how much memory do we need? We know that,

$$\pi(x) > \frac{x}{\ln(x)} > \frac{x}{\lg(x)}$$

Then let $x = 2^n$,

$$\frac{x}{\lg(x)} = \frac{2^n}{n} = \frac{2^n}{2^{\lg(n)}} = 2^{n-\lg(n)}$$

Taking this rough lower bound, there are at least $2^{32-5} = 2^{27}$ prime numbers we need to store. If we take 32 bits each,

$$\frac{2^{27} \times 32 \text{ bits}}{2^{20} \text{ megabits}} = 2^{27+5-20} = 2^{12} \text{ megabits} = \frac{2^{12} \text{ megabits}}{8 \text{ bits}} = 2^{12-3} = 2^9 = 512 \text{ megabytes.}$$

Can we do better? Yes, we can! We can write each odd prime number as a sum of prime-gaps to it. Take 7 for example,

$$7 = p_4 = (p_2 - p_1) + (p_3 - p_2) + (p_4 - p_3) + (p_5 - p_4) = (3-2) + (5-3) + (7-5) + (11-7)$$

In general,

$$p_j = \sum_{k=1}^{k=j} (p_{k+1} - p_k) \quad \Rightarrow \quad p_{j+1} = p_j + (p_{j+1} - p_j)$$

Therefore, $7 = 5 + 2$. The largest prime-gap between any consecutive primes p_j and p_{j+1}, i. e., $p_{j-1} - p_j$ for $p_j < p_{j+1} \leq p_i = 4294967291$ is known to be 336. If we half all the prime-gaps starting at $p_3 - p_2 = 5 - 3$ then we always get an integer. Hence, we may half 336 to 168 obtaining a number that fits in just 8 bits or a byte!

Instead of saving 32-bit prime numbers themselves, we can save halved distances between them starting at $p_3 - p_2 = 5 - 3$ and ending in $p_{203280221} - p_{203280220}$. Let's perform the above calculation again but this time taking only 8 bits.

$$\frac{2^{27} \times 8 \text{ bits}}{2^{20} \text{ megabits}} = 2^{27+3-20} = 2^{10} \text{ megabits} = \frac{2^{10} \text{ megabits}}{8 \text{ bits}} = 2^{10-3} = 2^7 = 128 \text{ megabytes.}$$

Not bad. But this is only a lower bound. Since we know that $\pi(2^{32}) = i = 203280221$, we can calculate the exact amount of storage.

$$(203280221 - 2) \times 2^{3-20-3} = 193.86312389373779296875 < 194 \text{ megabytes}$$

[1] $\pi(2^{32})$ is the *prime-counting function*: counting the number of primes less than or equal to 2^{32}.

1.2. 64-bit Prime Numbers

Since most machines these days are 64 bit, it makes sense to ask how much memory do we need to save all 64-bit primes. We utilise the same halved-gaps optimisation from the previous section. The largest prime-gap between two consecutive primes which are possible to store using 64-bits is known to be 1550. Half of it is 775 which fits 16 bits or 2 bytes. Let's use the same lower-bound on $\pi(2^{64})$,

$$\frac{2^{64-6} \times 16 \text{ bits}}{2^{50} \text{ petabits}} = 2^{58+4-50} = 2^{12} \text{ petabits} = \frac{2^{12} \text{ petabits}}{8 \text{ bits}} = 2^{12-3} = 2^9 = 512 \text{ petabytes}.$$

Which is a bit much.

1.3. 8-bit Addressing

One may wonder in case of 64-bit primes, if the largest half-gap we need to save is 775, i. e.,

$$\lfloor \lg(775) \rfloor + 1 = 10 \text{ bits}.$$

Then why are we using 2 bytes or 6 extra bits for each gap? This is because the smallest unit of addressable memory is a byte. Even if the operating system needs to save a single bit (say 1), it has to save it as a byte: `0000 0001`. Due to this, 10 bits can only be saved using 2 bytes or 16 bits. This situation is usually remedied with fancy bit-packing algorithms. Albeit, I am keeping it simple.

University of Oklahoma