AuW-u04-pg-bf

Aufgabe 1 – *Verifizierer*

Nehmen Sie an, dass die folgenden zwei Algorithmen gegeben sind:

- Ein Monte-Carlo-Algorithmus[1] A, der als Input eine Zahl $N \in \mathbb{N}$ nimmt, und der eine Zahl X im Intervall $[N, 2N]$ ausgibt, sodass X mit Wahrscheinlichkeit $p_N \geq 1/\log N$ eine Primzahl ist. Die Laufzeit von A sei $\mathcal{O}(\log N)$.
- Ein deterministischer Algorithmus V ("Verifizierer"), der als Input ein $x \in \mathbb{N}$ nimmt, und der in Zeit $\mathcal{O}((\log n)^{6.1})$ entscheidet, ob x eine Primzahl ist.

Konstruieren Sie einen Las-Vegas-Algorithmus, der als Input ein $N \in \mathbb{N}$ nimmt, und der eine Primzahl im Intervall $[N, 2N]$ ausgibt. Was ist die Laufzeit Ihres Algorithmus, wenn Sie eine Erfolgswahrscheinlichkeit von mindestens $1/2$ erreichen wollen?

Algorithm A
Input: $N \in \mathbb{N}$
Output: $X \in (N, 2N)$
Runtime: $\mathcal{O}(\log N)$
Probability: $p_N \geq \frac{1}{\log N}$

Algorithm V
Input: $x \in \mathbb{N}$
Output: true / false
Runtime: $\mathcal{O}((\log n)^{6.1})$

Algorithm L
Input: $N \in \mathbb{N}$
Output: $X \in (N, 2N)$
Runtime: $\mathcal{O}(?)$
Probability: $p_N \geq \frac{1}{2}$

> **intuition:**
>
> run A (monte carlo) many times and check with verifier until we have Pr["correct answer"] = 1/2

To construct a Las Vegas algorithm (L) with a success rate of $\frac{1}{2}$, we can repeatedly apply Algorithm A and then use Algorithm V to verify the result. We will keep iterating until we find a result that passes the verification with certainty.

Here's a sketch of the algorithm:

1. Use Algorithm A to generate a candidate prime number X in the range $(N, 2N)$.
2. Use Algorithm V to verify if X is prime. If it is, return X.
3. If Algorithm V indicates that X is not prime, repeat steps 1 and 2.

Let's denote the runtime of Algorithm A as (T_A) and the runtime of Algorithm V as (T_V). We will denote the number of times we need to repeat Algorithm A until we find a prime candidate as (k).

Since the success rate of Algorithm A is ($p_N \geq \frac{1}{\log N}$), the expected number of iterations (k) until we find a prime candidate is at most ($2 \log N$).

Therefore, the runtime of Algorithm L will be:

$$\begin{aligned}
\text{Runtime of L} &= k \times T_A + T_V \\
&= 2 \log N \times T_A + T_V \\
&= 2 \log N \times O(\log N) + O((\log N)^{6.1}) \\
&= O((\log N)^2) + O((\log N)^{6.1}) \\
&= O((\log N)^{6.1})
\end{aligned}$$

So, the runtime of Algorithm L is ($O((\log N)^{6.1})$), which is dominated by the runtime of Algorithm V.