To avoid large integer arithmetic, we can use an almost linear hash function $h(x) = x \bmod p$, where p is a random prime in $poly(n)$. Then $a + b = c$ iff $(h(a) + h(b)) \bmod p = h(c) \bmod p$ w.h.p. We can preprocess in $O(n)$ time by Rabin-Karp type of hashing, and query the hash value of any subinterval in $O(1)$ time. Let the numbers be $n_1, n_2, n_3, \ldots, n_t$, we can enumerate n_t (i.e. enumerate its starting point), we claim then n_{t-1} and n_{t-2} are fixed. This is because there are no leading zeros, and the sequence n_i is monotone increasing (except the first two terms), so if we let $\ell(x)$ denote the length of x, then $\ell(n_{t-1})$ equals to either $\ell(n_t)$ or $\ell(n_t) - 1$. To show these two cases cannot simultaneously happen, let x denote the number before n_t with length $\ell(x) = \ell(n_t) - 1$, and assume the digit before x is c, then $\cdots c + x = \cdots + cx = n_t$, i.e. $\cdots c = \cdots + c \cdot 10^{\ell(x)}$, and the only solution is $\underbrace{c\cdots c}_{\ell(x)+1} = \underbrace{c\cdots c}_{\ell(x)} + c\underbrace{0\cdots 0}_{\ell(x)}$, which violates monotonicity.

n_i grows exponentially fast, i.e. $\forall i,\ 2n_{i+1} \geq n_{i+2} = n_i + n_{i+1} \geq 2n_i$, therefore $n_{i+1} + O(1) \geq \ell(n_{i+2}) \geq n_i + \Omega(1)$. $\ell(n_t) = \ell(\max\{n_1, n_2\}) + \Theta(t)$ (we can also see this fact from the fibonacci-like linear recursive equation), and therefore $t = O(\sqrt{n})$, $\ell(n_t) = \Omega(\sqrt{n})$. Also, $t = O(\frac{n}{\ell(n_t) - c\cdot\sqrt{n}})$, where c is a constant. For $\ell(n_t) \geq c'\sqrt{n}$, we can write this as $t = O(\frac{n}{\ell(n_t)})$. Enumerate $\ell(n_t)$ from $\Omega(\sqrt{n})$ to n, and verify the solution takes $O(t) = O(\frac{n}{\ell(n_t)})$ time. This is because if we fix n_{i+1} and n_i, then n_{i-1} and $h(n_{i-1})$ are fixed, assuming $h(n_{i-1}) = y$, n_{i-1} starts at index j and ends at index k (j is unknown and k is known), and let $\hat{h}_i = h(a_1 a_2 \ldots a_i)$ be the Rabin-Karp hash value of the prefix of the array, then we have $h(n_{i-1}) \equiv \hat{h}_k - \hat{h}_{j-1} \cdot 10^{k-j+1} \equiv y \pmod p$, i.e. $\hat{h}_{j-1} \cdot 10^{-(j-1)} \equiv (\hat{h}_k - y) \cdot 10^{-k} \pmod p$. Therefore we can find n_{i-1} in $O(1)$ time. In total $O(n \log n)$ time.

remark. use the general formula of linear recurrence equation for $n_{i+2} = n_{i+1} + n_i$, we get $n_i = c_1 \cdot (\frac{1+\sqrt{5}}{2})^i + c_2 \cdot (\frac{1-\sqrt{5}}{2})^i$, where c_1 and c_2 satisfies
$\begin{cases} c_1 \cdot (\frac{1+\sqrt{5}}{2}) + c_2 \cdot (\frac{1-\sqrt{5}}{2}) = n_1, \\ c_1 \cdot (\frac{1+\sqrt{5}}{2})^2 + c_2 \cdot (\frac{1-\sqrt{5}}{2})^2 = n_2. \end{cases}$
And also matrix representation:

$$\begin{bmatrix} n_{i+2} \\ n_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^i \begin{bmatrix} n_2 \\ n_1 \end{bmatrix}.$$

References