1. (a) Suppose $n$ is very large, the worst case is when the right most $k$ digits are all 1, so an increment will change every digit, including the left most digit. Thus, the worst-case cost is $\Theta(k)$.

(b) We observe that not all digits change every time we increment the counter. In a sequence of $n$ increment operations (suppose starting from the minimum $(-1, -1, \ldots, -1)$), $b_0$ changes every time, $b_1$ changes $\lceil \frac{n}{3} \rceil$ times, $b_2$ changes $\lceil \frac{n}{3^2} \rceil$ times, so on and so forth. Thus, the total number of digit changes is

$$
\sum_{i=0}^{k} \left\lfloor \frac{n}{3^i} \right\rfloor \leq n \sum_{i=0}^{k} \frac{1}{3^i} = \frac{1}{2} (3 - \frac{1}{3^k})n < \frac{3n}{2}.
$$

Therefore, the total cost is $O(n)$ and the amortized cost is $O(n)/n = O(1)$.

(c) Without loss of generality, suppose we start the $n$ increment operations from $(0, 0, \ldots, 0)$. We charge an amortized cost of 3 dollars to change a digit from 0 to 1, one for the actual cost, the rest two placed on the digit as credit for future use. When a digit is changed from 1 to -1, we pay 1 dollar for the actual cost and still have 1 dollar left. When the digit is changed from -1 to 0, we pay another 1 dollar. Thus, we do not need to charge more money to set a digit from 1 to -1 or from -1 to 0. Each increment operation sets at most 1 digit from 0 to 1, so $n$ increment operations set at most $n$ digits from 0 to 1, incurring a total amortized cost of at most $3n$, which is $O(n)$. On average, the amortized cost of each increment operation is $O(1)$. An alternative method is to charge an amortized cost of 1.5 dollars to set a digit from -1 to 0 and 1.5 dollars to set it from 0 to 1.

(d) As the binary counter example, we define the potential $\Phi$ to be the number of 1s in the counter. Each increment operation changes a digit from -1 to 0 or from 0 to 1 (in line 7), and suppose the $i$-th increment operation resets $t_i$ digits from 1 to -1 (in the while loop). Then, the actual cost is $c_i = t_i + 1$. The change of potential, i.e., the change of the number of 1s in the counter, is $-t_i$ or $1 - t_i$, so $\Phi_i - \Phi_{i-1} \leq 1 - t_i$. Thus, the amortized cost is

$$
\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} \leq t_i + 1 + 1 - t_i = 2 = O(1).
$$

(e) (1) We prove the uniqueness of the representation by infinite descent. Assume to the contrary that some number has more than one representation; let $N$ be the least positive such number (we can assume it is positive because if not, negate each coefficient). The lowest “bit” is determined by $N \mod 3$ and hence is unique, so $[N/3]$ must also have more than one representation. However, $N$ was the least such number, which causes a contradiction. Therefore, every integer’s representation must be unique.

(2) Obviously, the minimum integer is represented by a counter with -1 in all digits and the maximum integer is represented by a counter with 1 in all digits, so

$$
-\sum_{i=0}^{k} 3^i \leq n \leq \sum_{i=0}^{k} 3^i.
$$

We have $\sum_{i=0}^{k} 3^i = \frac{3^{k+1} - 1}{2}$. Thus, the range of integers that can be represented by a $k$-place Bachet counter is

$$
[-\frac{3^{k+1} - 1}{2}, \frac{3^{k+1} - 1}{2}].
$$
All possible values of the k positions are used, each giving a different value in the range, so k positions of coefficients cannot represent a larger range.

2. **Accounting method.** It is known that the ratio of two successive Fibonacci numbers converges to the golden ratio, i.e.,
\[
\lim_{i \to \infty} \frac{F_i}{F_{i-1}} \approx 1.618.
\]
In fact, we only need \( \frac{F_i}{F_{i-1}} \geq 1.5 \). We charge an amortized cost of 4 dollars to insert or delete an element, use 1 dollar to pay for the actual cost, and save the rest 3 in the bank. W.l.o.g., suppose we start from when the table size \( s = F_i \) and the number of elements \( n = F_{i-1} \).

1. We are able to make at least \( F_i - F_{i-1} \) insertions before incurring a new (larger) table creation and reinsertions of \( F_i \) elements. By then, we will have saved \( 3 \cdot (F_i - F_{i-1}) \) dollars, which is enough to cover the cost of reinsertions because
\[
\frac{F_i}{F_{i-1}} \geq 1.5 \Rightarrow 3 \cdot (F_i - F_{i-1}) \geq F_i. \tag{4}
\]
Then, after the table creation and reinsertions, there might be a little balance in the bank, which is fine.

2. Or, we are able to make at least \( F_{i-1} - F_{i-2} \) deletions before incurring the new (smaller) table creation and reinsertions of \( F_{i-2} \) elements. By then, we will have saved \( 3 \cdot (F_{i-1} - F_{i-2}) \) dollars, which is enough to cover the cost of reinsertions because
\[
\frac{F_{i-1}}{F_{i-2}} > 1.33 \Rightarrow 3 \cdot (F_{i-1} - F_{i-2}) > F_{i-2}. \tag{5}
\]
Therefore, we do not need to charge extra for reinsertions. A sequence of \( n \) insertions/deletions have a total amortized cost of at most \( 4n \), which is \( O(n) \). On average, the amortized cost of each operation is \( O(1) \).