AD Aflevering 2

zts900, chs380

February 2025

1.Amortiseret analyse

a)

We examine binary numbers where we increment. For example, we can observe the following:

Decimal	Binary	Number of 1s, $O(i)$
0	0000	0
1	0001	1
2	0010	1
3	0011	2

We want to find out how $\Phi(D_i)$ changes as i increases. If we look at the table, we observe that $\Phi(D_i)$ grows slowly and never exceeds the number of bits in i. A number with i bits has at most $\log i$ - ones, because there is no more than $O(\log i)$ bits in a binary representation.
At the same time, ut has at least $\Omega(log\ i)$ -ones in avarerage over a lot of numbers.
Therefore we have the following

$$\Omega(log\ i) \leq \Phi(D_i) \leq O(\log i)$$

This leads to the following conclusions:

1. Yes, because there er no more than $log\ i$ -ones in average
2. No, $\Phi(D_i)$ grows slower than i
3. No because $\Phi(D_i)$ grows, so it cannot be constant.
4. Yes, since there are at most $O(\log i)$ -ones in a number with i bits.
5. Yes, everything is $O(i)$, but $O(\log i)$ is a much tighter bound.

b)

We now consider incrementing a base-10 number:

- If the last digit is 9, it is set to 0, and the previous digit is incremented
- If it is not 9, the digit is simply incremented
- The number of affected digits varies from 0 to the total number of digits.

We therefore define the potential function as the number of digits that are 9 in the number:

$$\Phi(x_i) = \text{Number of digits that are 9 in } x_i$$

The real cost c_i for an increment operation depends on how many digits changes:
- If the last digit is not 9, the operation costs $O(1)$.
- If there are multiple nines, they get changed and updates a new non-nine digit

This costs t_{i+1}, where t_i is the number of changed digits.
Therefore, the potential change is:

$$\Phi(x_i) - \Phi(x_{i-1}) \leq (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i$$

The amortized cost is calculated as:

$$\hat{c}_i = c_i + \Phi(x_i) - \Phi(x_{i-1})$$

Since $c_i = t + 1$, we get:

$$\hat{c}_i \leq (t_i + 1) + (1 - t_i) = 2$$

which shows that the amortized time for incrementing is $O(1)$.

c)

We know that when we decrement in a base 10 the following happens:

- If the last digit is not 0, it is reduced by 1
- If the last digit is 0, it changes to 9, and we subtract 1 from the previous digit
- If multiple digits are 0, there would then happen multiple changes

The worst-case scenario occurs when all digits are 0, requiring up to k changes.

We define the potential function as the number of digits that are 0 in the counter:

$$\Phi(x_i) = (\text{number of digits that are 0 in } x_i)$$

The real cost c_i of a decrement operation depends on how many digits change:
- If the last digit is not 0, the cost is 1.
- If there is a sequence of 0, they reset to 9, and the previous digit decreases by 1, this costs $t_i + 1$, where t_i is the number of reset digits.

The potential difference is:

$$\Phi(x_i) - \Phi(x_{i-1}) \leq (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i$$

The amortized cost is then:

$$\hat{c_i} = c_i + \Phi(x_i) - \Phi(x_{i-1})$$

Since $c_i = t_i + 1$, we get:

$$\hat{c_i} \leq (t_i + 1) + (1 - t_i) = 2$$

In the worst case, the entire counter consists of 0's, requiring up to k changes, so we get an upper bound of $O(k)$.

d)

We have already shown that increment can be performed in amortized time $O(1)$.
We have also shown that decrement can be performed in amortized time $O(k)$.
We can force the counter to take $O(k)$ time per operation by alternating between increment and decrement in a specific sequence.

- Increment up to 999...9 (all the digits are 9) : This takes $O(k)$ time, but amortized over many operations, it is still $O(1)$.
- Decrement from 999...9 to 000...0: This requires k changes, as all digits switch from 9 to 0. - This takes $O(k)$ time.
- Increment from 000...0 to 999...9 again: As before, this takes $O(k)$ time.
- If we repeat the above operations frequently, the average cost per operation is no longer $O(1)$, but instead $O(k)$.

This sequence shows that we cannot have a constant amortized bound, independent of k, for both Increment and Decrement.
If we force the counter to oscillate between 999...9 and 000...0, we get a situation where both operations require $O(k)$ in the worst case. This means it is impossible to guarantee $O(1)$ amortized time for both operations simultaneously.

2. Fibonacci Heap

We analyze the runtime and structure of a Fibonacci heap given:

- n_1 insert operations
- n_2 extract-min operations
- n_3 decrease-key operations

The total number of operations is $n = n_1 + n_2 + n_3$.

Operation	Amortized Time	Worst-case Time
Insert(x)	$O(1)$	$O(1)$
Extract-min(x)	$O(\log n)$	$O(n)$
Decrease-key(x)	$O(1)$	$O(1)$

Table 1: Time Complexity of Fibonacci Heap Operations

1. No, insert and decrease-key have a worst-case time of $O(1)$, while extract-min has $O(n)$ in the worst case. Thus, the worst-case complexity is $O(n)$
2. Yes, because the dominant operation is extract-min, which takes $O(n)$ in the worst case.
3. Yes, because insert occurs n_1 times, each operation taking $O(1)$, resulting in a total time of $O(n_1)$. Extract-min occurs n_2 times, each taking $O(\log n)$, leading to a total time of $O(n_2 \log n)$. Decrease-key occurs n_3 times, each taking $O(1)$, therefore $O(n_3)$. Thus, the total runtime is:

$$O(n_1) + O(n_3) + O(n_2 \log n) = O(n_1 + n_3 + n_2 \log n).$$

4. Yes, insert increases the number of nodes, whereas each extract-min gets rid of a node. Decrease-key does not directly affect the number of nodes.
5. No, because decrease-key only moves nodes when their parent has already lost half of its children. This affects the total number of decrease-key operations, but the sum $n_1 + n_3$ is not necessarily well-defined.

3 Correctness arguments

We wand to select a partial set $I^* \subseteq \{1, ...n\}$from a list of n numbers that gives the highest product.

1. True, even if we only have one element and that element is between 0 and 1 we get a higher product = 1 for the empty set.
2. False, we only want an even number of negative indexes so that they cancel, so if a uneven number is present in the list not all will be selected.
3. True, mulitplying with $x_i > 1$ always gives a larger product
4. False, for same reason as 2 since this case also includes negatives

---

4 Induction

We assume: $1 \leq p \leq r \leq n$:

B.s

$low = p$, $high = r + 1$, $i = 0$ gives

$$high_0 - low_0 = r + 1 - p \leq n + 1 - p \leq n + 1 - 1 = n = \frac{n}{2^0}$$

Holds

I.h

$$H_i - L_i \leq \frac{n}{2^i}$$

Variations:

$$H_i \leq \frac{n}{2^i} + L_i$$

$$H_i - \frac{n}{2^i} \leq L_i$$

$$-H_i + \frac{n}{2^i} \geq L_i$$

I.s

We want to prove: $high_{i+1} - low_{i+1} \leq \frac{n}{2^{i+1}}$.
Case1 $x \leq T[mid_{i-1}]$:

$$H_{i+1} = \left\lfloor \frac{L_i + H_i}{2} \right\rfloor$$

$$L_{i+1} = L_i$$

Gives:

$$\begin{aligned}
H_{i+1} - L_{i+1} &= \left\lfloor \frac{L_i + H_i}{2} \right\rfloor - L_i \\
&\leq \left\lfloor \frac{L_i + \frac{n}{2^{\mathrm{i}}} + L_i}{2} \right\rfloor - L_i \\
&= \left\lfloor \frac{2L_i + \frac{n}{2^{\mathrm{i}}}}{2} \right\rfloor - L_i \\
&= \left\lfloor L_i + \frac{n}{2^{\mathrm{i}+1}} \right\rfloor - L_i \\
&= \left\lfloor \frac{n}{2^{\mathrm{i}+1}} \right\rfloor + L_i - L_i \\
&= \left\lfloor \frac{n}{2^{\mathrm{i}+1}} \right\rfloor \\
&\leq \frac{n}{2^{\mathrm{i}+1}}
\end{aligned}$$

Case2 $x > T[mid_{i-1}]$:

$$L_{i+1} = \left\lfloor \frac{L_i + H_i}{2} \right\rfloor + 1$$

$$H_{i+1} = H_i$$

Gives:

$$\begin{aligned}
H_{i+1} - L_{i+1} = H_i - (\left\lfloor \frac{L_i + H_i}{2} \right\rfloor + 1) \\
\leq & H_i - (\left\lfloor \frac{L_i + \frac{n}{2^{\mathrm{i}}} + L_i}{2} \right\rfloor + 1) \\
= & H_i - (\left\lfloor \frac{2L_i + \frac{n}{2^{\mathrm{i}}}}{2} \right\rfloor + 1) \\
= & H_i - \left\lfloor L_i + \frac{n}{2^{\mathrm{i}+1}} \right\rfloor - 1 \\
\leq & \frac{n}{2^{\mathrm{i}}} + L_i - \left\lfloor L_i + \frac{n}{2^{\mathrm{i}+1}} \right\rfloor - 1 \\
= & \frac{n}{2^{\mathrm{i}}} + L_i - L_i + \left\lfloor \frac{n}{2^{\mathrm{i}+1}} \right\rfloor - 1 \\
\leq & \frac{n}{2^{\mathrm{i}}} + L_i - L_i + \frac{n}{2^{\mathrm{i}+1}} + 1 - 1 \\
= & \frac{n}{2^{\mathrm{i}}} + \frac{n}{2^{\mathrm{i}+1}}
\end{aligned}$$

2 Trying again:

$$
\begin{aligned}
H_{i+1} - L_{i+1} = H_i - (\left\lfloor \frac{L_i + H_i}{2} \right\rfloor + 1) \\
\leq & H_i - (\left\lfloor \frac{-H_i + \frac{n}{2^{\mathrm{i}}} + H_i}{2} \right\rfloor + 1) \\
= & H_i - (\left\lfloor \frac{\frac{n}{2^{\mathrm{i}}}}{2} \right\rfloor + 1) \\
= & H_i - (\left\lfloor \frac{n}{2^{\mathrm{i}+1}} \right\rfloor + 1) \\
\leq & H_i - (\frac{n}{2^{\mathrm{i}+1}} + 1 + 1) \\
= & H_i - \frac{n}{2^{\mathrm{i}+1}} - 2 \\
\leq & H_i - L_i - \frac{n}{2^{\mathrm{i}}} * \frac{1}{2} - 2 + L_i \\
\leq & \frac{n}{2^{\mathrm{i}}} - \frac{n}{2^{\mathrm{i}}} * \frac{1}{2} - 2 + L_i \\
= & \frac{n}{2^{\mathrm{i}}}(1 - \frac{1}{2}) - 2 + L_i \\
= & \frac{n}{2^{\mathrm{i}+1}} - 2 + L_i
\end{aligned}
$$

I give up, give me a hint