

Chapter 17 Solusion

<https://github.com/frc123/CLRS>

12/28/2021

17.1

17.1-1

No. Consider we operate `MULTPUSH(S, n)` n times. Such n operations cost $\Theta(n^2)$, so the amortized cost is $\Theta(n)$.

Actually, we can `MULTPUSH` incredible large amount of items, so $O(1)$ of course cannot be bound on the amortized cost of stack operations.

17.1-2

Consider a k -bit counter where each bit in the counter is 1. Now, we perform `INCREMENT` which flips $k + 1$ bits. Then, we perform `DECREMENT` which flips $k + 1$ bits again. Hence perform a sequence of length n operations $\langle \text{INCREMENT}, \text{DECREMENT}, \text{INCREMENT}, \text{DECREMENT}, \dots \rangle$ cost $\Theta(nk)$ in total.

17.1-3

$$n + \sum_{i=1}^{\lfloor \lg n \rfloor} (2^i - 1) \leq n + \sum_{i=0}^{\lg n} 2^i = n + 2^{\lg n + 1} - 1 = n + 2n - 1 = 3n - 1$$

Hence the amortized cost per operation is $O(1)$.

17.2

17.2-1

operation	actual cost	amortized cost
PUSH	1	2
POP	1	2
Copy	s	0

where s is the stack size when it is called which has an upper bound k .

Each operation (PUSH or POP) charges an amortized cost of 2 and actual use 1. After k operations, we have k credits, and copy operation cost at most k . Hence we conclude the total amortized cost is greater than the total actual cost at all times.

17.2-2

Let the amortized cost of each operation be 3. We want to show that

$$\sum_{i=1}^n \hat{c}_i \geq \sum_{i=1}^n c_i$$

for all integers n where

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2,} \\ 1 & \text{otherwise} \end{cases}$$

and $\hat{c}_i = 3$ for all integers i . That is we want to show that

$$3n \geq n + \sum_{i=1}^{\lfloor \lg n \rfloor} (2^i - 1).$$

By exercise 17.1-3, we have

$$n + \sum_{i=1}^{\lfloor \lg n \rfloor} (2^i - 1) \leq 3n - 1.$$

Hence the amortized cost per operation is $O(1)$.

17.2-3

As the hint mentioned, we keep a pointer to the high-order 1 and maintain it during the operations. In each INCREMENT operation, we check if the high-order 1 moved to a higher order.

Flipping a bit charges 1. Moving the pointer to the high-order 1 charges \$1. Let the amortized cost of each INCREMENT operation be \$4, and let the amortized cost of each RESET operation be \$1. When we set a bit to 1, we actually cost \$1 and retain \$2 as credits for the purpose of setting to 0 and resetting. If we need to update pointer, we charge another \$1. Hence amortized cost of each INCREMENT operation is \$4. Each RESET operation need to move the pointer to -1 , so it costs \$1.

```
1  struct Counter
2  {
3      int length;
4      std::vector<bool> bits;
5      int high_order_one;
6
7      Counter(int length) : length(length),
8                          bits(length, 0), high_order_one(-1) {}
9  };
```

```
10
11 void Increment(Counter& counter)
12 {
13     int i;
14     i = 0;
15     while (i < counter.length && counter.bits[i] == 1)
16     {
17         counter.bits[i] = 0;
18         ++i;
19     }
20     if (i < counter.length)
21     {
22         counter.bits[i] = 1;
23         counter.high_order_one = std::max(i, counter.high_order_one);
24     }
25     else
26     {
27         // overflow
28         counter.high_order_one = -1;
29     }
30 }
31
32 void Reset(Counter& counter)
33 {
34     int i;
35     for (i = 0; i < counter.length; ++i)
36     {
37         counter.bits[i] = 0;
38     }
39     counter.high_order_one = -1;
40 }
```

17.3

17.3-1

Let $\Phi'(D_i) = \Phi(D_i) - \Phi(D_0)$. Clearly, $\Phi'(D_0) = 0$. We claim the amortized costs using Φ' are the same as the amortized costs using Φ .

$$\begin{aligned}\hat{c}_i &= c_i + \Phi'(D_i) - \Phi(D_{i-1}) \\ &= c_i + (\Phi(D_i) - \Phi(D_0)) - (\Phi(D_{i-1}) - \Phi(D_0)) \\ &= c_i + \Phi(D_i) - \Phi(D_{i-1})\end{aligned}$$

17.3-2

Let $\Phi(D_0) = 0$ and $\Phi(D_i) = 2(i - 2^{\lfloor \lg i \rfloor})$ for $i \geq 1$.

$$\begin{aligned}\Phi(D_i) - \Phi(D_{i-1}) &= 2(i - 2^{\lfloor \lg i \rfloor}) - 2((i-1) - 2^{\lfloor \lg(i-1) \rfloor}) \\ &= 2 - 2(2^{\lfloor \lg i \rfloor} - 2^{\lfloor \lg(i-1) \rfloor})\end{aligned}$$

Note that

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2,} \\ 1 & \text{otherwise} \end{cases}$$

Case 1. i is an exact power of 2.

$$\begin{aligned}\Phi(D_i) - \Phi(D_{i-1}) &= 2 - 2(i - \frac{i}{2}) \\ &= 2 - i\end{aligned}$$

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= i + 2 - i \\ &= 2\end{aligned}$$

Case 2. i is not an exact power of 2.

Then $2^{\lfloor \lg i \rfloor} = 2^{\lfloor \lg(i-1) \rfloor}$.

$$\Phi(D_i) - \Phi(D_{i-1}) = 2$$

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= 1 + 2 \\ &= 3\end{aligned}$$

Hence the amortized cost per operation is $O(1)$.

17.3-3

The idea is to let the potential be proportional to the sum of the height of every node in the min-heap. Note that an binary heap is a complete binary tree.

$$\sum_{j=1}^n \lfloor \lg j \rfloor \leq \lg(n!) \leq n \lg n$$

Let Φ be

$$\Phi(D_i) = \begin{cases} 0 & \text{if } n_i = 0, \\ kn_i \lg n_i & \text{if } n_i > 0 \end{cases}$$

for some constant k where n_i is the number of nodes in D_i . Also, we have

$$c_i \leq \begin{cases} k_1 \lg n_i & \text{if INSERT is performed in the } i\text{th operation and } n_i \geq 2, \\ k_2 \lg n_{i-1} & \text{if EXTRACT-MIN is performed in the } i\text{th operation and } n_{i-1} \geq 2 \end{cases}$$

Let $k = \max(k_1, k_2)$.

Case 1. INSERT is performed in the i th operation. Then $n_i - 1 = n_{i-1}$.

If $n_i = 1$,

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= c_i \end{aligned}$$

If $n_i \geq 2$,

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &\leq k \lg n_i + kn_i \lg n_i - kn_{i-1} \lg n_{i-1} \\ &= k(\lg n_i + n_i \lg n_i - n_{i-1} \lg n_{i-1}) \\ &= k(\lg n_i + n_i \lg n_i - (n_i - 1) \lg(n_i - 1)) \\ &= k(\lg n_i + n_i \lg n_i - n_i \lg(n_i - 1) + \lg(n_i - 1)) \\ &< k(2 \lg n_i + n_i(\lg n_i - \lg(n_i - 1))) \end{aligned}$$

Note that $\forall x \in \mathbb{R}, 1 + x \leq e^x$. Then

$$\begin{aligned} n_i(\lg n_i - \lg(n_i - 1)) &= n_i \lg \frac{n_i}{n_i - 1} \\ &= n_i \lg \left(1 + \frac{1}{n_i - 1}\right) \\ &\leq n_i \lg \left(e^{\frac{1}{n_i - 1}}\right) \\ &= \frac{n_i}{n_i - 1} \lg e \\ &= \left(1 + \frac{1}{n_i - 1}\right) \lg e \\ &\leq 2 \lg e \end{aligned}$$

Hence

$$\hat{c}_i < k(2 \lg n_i + 2 \lg e)$$

We conclude $\hat{c}_i = O(\lg n)$ for INSERT.

Case 2. EXTRACT-MIN is performed in the i th operation. Then $n_{i-1} - 1 = n_i$.

If $n_{i-1} = 1$,

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= c_i\end{aligned}$$

If $n_{i-1} \geq 2$,

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &\leq k \lg n_{i-1} + kn_i \lg n_i - kn_{i-1} \lg n_{i-1} \\ &= k(\lg n_{i-1} + n_i \lg n_i - n_{i-1} \lg n_{i-1}) \\ &= k(\lg n_{i-1} + (n_{i-1} - 1) \lg(n_{i-1} - 1) - n_{i-1} \lg n_{i-1}) \\ &< k(\lg n_{i-1} - \lg(n_{i-1} - 1)) \\ &= k \lg\left(1 + \frac{1}{n_{i-1} - 1}\right) \\ &\leq k \lg e^{\frac{1}{n_{i-1} - 1}} \\ &= \frac{k}{n_{i-1} - 1} \lg e\end{aligned}$$

We conclude $\hat{c}_i = O(1)$ for EXTRACT-MIN.

17.3-4

$$\Phi(D_n) - \Phi(D_0) = s_n - s_0$$

Since $\hat{c}_i = 2$,

$$\begin{aligned}\sum_{i=1}^n c_i &= \sum_{i=1}^n \hat{c}_i - \Phi(D_n) + \Phi(D_0) \\ &= 2n + s_0 - s_n\end{aligned}$$

17.3-5

$$\Phi(D_0) = b$$

Since $\hat{c}_i \leq 2$,

$$\begin{aligned}\sum_{i=1}^n c_i &= \sum_{i=1}^n \hat{c}_i - \Phi(D_n) + \Phi(D_0) \\ &\leq 2n + b - \Phi(D_n)\end{aligned}$$

Since $\Phi(D_n) \geq 0$,

$$\sum_{i=1}^n c_i \leq 2n + b$$

Since $n = \Omega(b)$,

$$\sum_{i=1}^n c_i = O(n)$$

17.3-6

```
1  template <typename T>
2  class Queue
3  {
4  public:
5      void Enqueue(T& x);
6      void Enqueue(T&& x);
7      T Dequeue();
8  private:
9      std::stack<T> s_a_;
10     std::stack<T> s_b_;
11 };
12
13 template <typename T>
14 void Queue<T>::Enqueue(T& x)
15 {
16     s_a_.push(x);
17 }
18
19 template <typename T>
20 void Queue<T>::Enqueue(T&& x)
21 {
22     s_a_.emplace(std::move(x));
23 }
24
25 template <typename T>
26 T Queue<T>::Dequeue()
27 {
28     if (s_b_.empty())
29     {
30         while (s_a_.empty() == false)
31         {
32             s_b_.emplace(std::move(s_a_.top()));
33             s_a_.pop();
34         }
35     }
```

```
36     T top = std::move(s_b_.top());
37     s_b_.pop();
38     return std::move(top);
39 }
```

Assume each of *s_a.push* (or *emplace*), *s_a.pop*, *s_b.push* (or *emplace*), *s_b.pop* costs \$1.
Then

$$c_i = \begin{cases} 1 & \text{if ENQUEUE is performed in the } i\text{th operation,} \\ 1 & \text{if DEQUEUE is performed in the } i\text{th operation and } D_{i-1}.s_b_ \text{ is not empty,} \\ 2 \cdot (D_{i-1}.s_a_size()) + 1 & \text{if DEQUEUE is performed in the } i\text{th operation and } D_{i-1}.s_b_ \text{ is empty} \end{cases}$$

Let

$$\Phi(D_i) = 3 \cdot (D_i.s_a_size()) + (D_i.s_b_size())$$

Case 1. ENQUEUE is performed in the *i*th operation.

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= 1 + 3 \cdot (D_i.s_a_size() - D_{i-1}.s_a_size()) + (D_i.s_b_size() - D_{i-1}.s_b_size()) \\ &= 1 + 3 \cdot 1 + 0 \\ &= 4 \end{aligned}$$

Case 2. DEQUEUE is performed in the *i*th operation and *D_{i-1}.s_b_* is not empty.

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= 1 + 3 \cdot (D_i.s_a_size() - D_{i-1}.s_a_size()) + (D_i.s_b_size() - D_{i-1}.s_b_size()) \\ &= 1 + 3 \cdot 0 - 1 \\ &= 0 \end{aligned}$$

Case 3. DEQUEUE is performed in the *i*th operation and *D_{i-1}.s_b_* is empty.

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= (2 \cdot (D_{i-1}.s_a_size()) + 1) + 3 \cdot (D_i.s_a_size() - D_{i-1}.s_a_size()) + (D_i.s_b_size() - D_{i-1}.s_b_size()) \\ &= (2 \cdot (D_{i-1}.s_a_size()) + 1) - 3 \cdot (D_{i-1}.s_a_size()) + (D_{i-1}.s_a_size() - 1) \\ &= 0 \end{aligned}$$

Thus, we conclude that the amortized cost of each ENQUEUE and each DEQUEUE operation is $O(1)$.

17.3-7

Note that section 9.3 provides an approach of selection in worst-case linear time.


```
1  class DataStructure
2  {
3  public:
4      void Insert(int x);
5      void DeleteLargerHalf();
6      const std::vector<int>& Get() const;
7  private:
8      std::vector<int> arr_;
9  };
10
11 void DataStructure::Insert(int x)
12 {
13     arr_.push_back(x);
14 }
15
16 void DataStructure::DeleteLargerHalf()
17 {
18     size_t median = arr_.size() >> 1;
19     LinearSelect(arr_, 0, arr_.size() - 1, (arr_.size() - 1) >> 1);
20     arr_.erase(arr_.begin() + median, arr_.end());
21 }
22
23 const std::vector<int>& DataStructure::Get() const
24 {
25     return arr_;
26 }
```

Assume

$$c_i = \begin{cases} 1 & \text{if INSERT is performed in the } i\text{th operation,} \\ n_{i-1} & \text{if DELETE-LARGER-HALF is performed in the } i\text{th operation} \end{cases}$$

where n_i is $|S|$ after the i th operation. Let

$$\Phi(D_i) = 2n_i$$

be the potential function of the data structure.

Case 1. INSERT is performed in the i th operation.

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= c_i + 2(n_i - n_{i-1}) \\ &= 1 + 2 \cdot 1 \\ &= 3 \end{aligned}$$

Case 2. DELETE-LARGER-HALF is performed in the i th operation.

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= c_i + 2(n_i - n_{i-1}) \\ &= n_{i-1} + 2\left(\frac{n_{i-1}}{2} - n_{i-1}\right) \\ &= n_{i-1} - n_{i-1} \\ &= 0\end{aligned}$$

17.4

17.4-1

By Theorem 11.6 and Theorem 11.8, assuming uniform hashing, for $\alpha < 1$, we know the expected number of probes in an unsuccessful search is at most $\frac{1}{1-\alpha}$ and in a successful search is at most $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$.

$$\lim_{\alpha \rightarrow 1^-} \frac{1}{1-\alpha} = \infty$$

$$\lim_{\alpha \rightarrow 1^-} \frac{1}{\alpha} \ln \frac{1}{1-\alpha} = \infty$$

Actually, when $\alpha = 1$, an unsuccessful search costs $\Theta(m) = \Theta(n)$. If we can bound α above by some constant that is strictly less than 1, the expected time of an unsuccessful or successful search is bounded above by some constant also.

Consider the function

$$\Phi_i = 2 \cdot \text{num}_i - \beta \cdot \text{size}_i$$

Let $\Phi_0 = 0$.

We just need to simply modify the conditional statement in line 4 of TABLE-INSERT to

if $T.\text{num} + 1 > \beta \cdot T.\text{size}$

where β is some constant that is strictly less than 1 and modify the base case of resizing table by modifying line 3 of TABLE-INSERT to

if $T.\text{size} = \lceil \frac{1}{\beta} \rceil$

We want to show that one expansion (twice the size) is enough in order to insert an element into a full table ($T.\text{num} + 1 > \beta \cdot T.\text{size}$).

Lemma 1. Assume $\text{num} \geq 1$. $\text{num} \leq \beta \cdot \text{size} \implies \text{num} + 1 \leq 2\beta \cdot \text{size}$

Proof. Note that $\forall x \geq 1, x + 1 \leq 2x$.

$$\begin{aligned}\text{num} \leq \beta \cdot \text{size} &\iff 2 \cdot \text{num} \leq 2\beta \cdot \text{size} \\ &\iff \text{num} + 1 \leq 2 \cdot \text{num} \leq 2\beta \cdot \text{size}\end{aligned}$$

□

Claim 2. $num_i \leq \beta \cdot size_i$ for all i .

Proof. We prove by induction. WLOG, assume inserts are performed for all i . Then $num_{i+1} = i+1$.

(Base) $k = 0$: $num_0 = 0 \leq \beta \cdot 0 = \beta \cdot size_0$

$k = 1$: $num_1 = 1 \leq \beta \cdot \lceil \frac{1}{\beta} \rceil = \beta \cdot size_1$ since $x \cdot \lceil \frac{1}{x} \rceil \geq x \cdot \frac{1}{x} = 1$ for all $x > 0$.

(Induction) Fix $k \geq 1$. Suppose that $num_k \leq \beta \cdot size_k$.

Case 1. $num_k + 1 \leq \beta \cdot size_k$

Then $size_{k+1} = size_k$.

$$num_{k+1} = num_k + 1 \leq \beta \cdot size_k = \beta \cdot size_{k+1}$$

Case 2. $num_k + 1 > \beta \cdot size_k$

Then $size_{k+1} = 2 \cdot size_k$. By lemma 1 and inductive hypothesis, we have $num_k + 1 \leq 2\beta \cdot size_k$.

$$num_{k+1} = num_k + 1 \leq 2\beta \cdot size_k = \beta \cdot size_{k+1}$$

□

We also want to show that $\Phi(T)$ is always nonnegative.

Claim 3. $\Phi_i \geq 0$ for all i

Proof. We prove by induction. WLOG, assume inserts are performed for all i . Then $num_{i+1} = i+1$.

(Base)

$$\Phi_0 = 0$$

$$\Phi_1 = 2 \cdot num_1 - \beta \cdot size_1 = 2 \cdot 1 - \beta \cdot \lceil \frac{1}{\beta} \rceil > 2 - \beta \cdot (\frac{1}{\beta} + 1) = 1 - \beta > 0$$

(Induction) Fix $k \geq 1$. Suppose that $\Phi_k = 2 \cdot num_k - \beta \cdot size_k \geq 0$.

Case 1. $num_k + 1 \leq \beta \cdot size_k$

Then $size_{k+1} = size_k$.

$$\Phi_{k+1} = 2 \cdot num_{k+1} - \beta \cdot size_{k+1} = 2 \cdot (num_k + 1) - \beta \cdot size_k > 2 \cdot num_k - \beta \cdot size_k \stackrel{\text{IH}}{\geq} 0$$

Case 2. $num_k + 1 > \beta \cdot size_k$

Then $size_{k+1} = 2 \cdot size_k$.

$$\Phi_{k+1} = 2 \cdot num_{k+1} - \beta \cdot size_{k+1} = 2 \cdot (num_k + 1) - 2\beta \cdot size_k = 2 \cdot (num_k + 1 - \beta \cdot size_k) > 0$$

Since $num_k + 1 - \beta \cdot size_k > 0$

□

We want to analysis the expected amortized cost. By Theorem 11.6, we assume

$$E[c_i] = \begin{cases} 1 & \text{if the } i\text{th insert operation does not trigger an expansion,} \\ num_i & \text{if the } i\text{th insert operation does trigger an expansion} \end{cases}$$

If the i th insert operation does not trigger an expansion, then we have $size_i = size_{i-1}$, and the expected amortized cost of the operation is

$$\begin{aligned} E[\hat{c}_i] &= E[c_i] + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \cdot num_i - \beta \cdot size_i) - (2 \cdot num_{i-1} - \beta \cdot size_{i-1}) \\ &= 1 + (2 \cdot num_i - \beta \cdot size_i) - (2 \cdot (num_i - 1) - \beta \cdot size_i) \\ &= 3. \end{aligned}$$

If the i th insert operation does trigger an expansion, then we have $size_i = 2 \cdot size_{i-1}$, and the expected amortized cost of the operation is

$$\begin{aligned} E[\hat{c}_i] &= E[c_i] + \Phi_i - \Phi_{i-1} \\ &= num_i + (2 \cdot num_i - \beta \cdot size_i) - (2 \cdot num_{i-1} - \beta \cdot size_{i-1}) \\ &= (num_{i-1} + 1) + (2 \cdot (num_{i-1} + 1) - \beta \cdot 2 \cdot size_{i-1}) - (2 \cdot num_{i-1} - \beta \cdot size_{i-1}) \\ &= 3 + num_{i-1} - \beta \cdot size_{i-1} \\ &< 3. \quad (\text{by claim 2}) \end{aligned}$$

Note that $E[c_i] = num_i$, which is linear, in this case because we need to copy all num_{i-1} elements to the new allocated table.

17.4-2

Claim 4. $\forall num \geq 2, \frac{num}{size} \geq \frac{1}{2} \implies \frac{num-1}{size} \geq \frac{1}{4}$

Proof.

$$\begin{aligned} \frac{num}{size} \geq \frac{1}{2} &\iff 2 \cdot num \geq size \\ &\iff 4(num - 1) \geq 2 \cdot num \geq size \quad (\text{since } num \geq 2) \\ &\iff \frac{num - 1}{size} \geq \frac{1}{4} \end{aligned}$$

□

Suppose that $\alpha_{i-1} \geq \frac{1}{2}$ and the i th operation is TABLE-DELETE. Then $num_i = num_{i-1} - 1$. By the claim, we know that a contraction will not be triggered in the i th operation if $\alpha_{i-1} \geq \frac{1}{2}$ and $num_{i-1} \geq 2$. If $num_{i-1} = 1$, then $num_i = 0$, which is trivial. Assume $num_{i-1} \geq 2$ in the following analysis. Then $c_i = 1$ and $size_i = size_{i-1}$.

Case 1. $\alpha_i < \frac{1}{2}$.

$$\begin{aligned}
 \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
 &= 1 + \left(\frac{\text{size}_i}{2} - \text{num}_i\right) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1}) \\
 &= 1 + \left(\frac{\text{size}_{i-1}}{2} - (\text{num}_{i-1} - 1)\right) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1}) \\
 &= 2 + \frac{3}{2} \cdot \text{size}_{i-1} - 3 \cdot \text{num}_{i-1} \\
 &= 2 + \frac{3}{2} \cdot \text{size}_{i-1} - 3\alpha_{i-1} \cdot \text{size}_{i-1} \\
 &\leq 2 + \frac{3}{2} \cdot \text{size}_{i-1} - \frac{3}{2} \cdot \text{size}_{i-1} \\
 &= 2
 \end{aligned}$$

Case 2. $\alpha_i \geq \frac{1}{2}$.

$$\begin{aligned}
 \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
 &= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1}) \\
 &= 1 + (2 \cdot (\text{num}_{i-1} - 1) - \text{size}_{i-1}) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1}) \\
 &= -1
 \end{aligned}$$

17.4-3

Suppose that TABLE-DELETE is performed in the i th operation. Then $\text{num}_i = \text{num}_{i-1} - 1$.

Case 1. $\frac{\text{num}_{i-1}-1}{\text{size}_{i-1}} \geq \frac{1}{3}$ (a contraction is not triggered in the i th operation).

Then we have $c_i = 1$ and $\text{size}_i = \text{size}_{i-1}$.

$$\begin{aligned}
 \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
 &= c_i + |2 \cdot \text{num}_i - \text{size}_i| - |2 \cdot \text{num}_{i-1} - \text{size}_{i-1}| \\
 &= 1 + |2 \cdot (\text{num}_{i-1} - 1) - \text{size}_{i-1}| - |2 \cdot \text{num}_{i-1} - \text{size}_{i-1}| \\
 &\stackrel{\Delta}{\leq} 1 + (|2 \cdot \text{num}_{i-1} - \text{size}_{i-1}| + |-2|) - |2 \cdot \text{num}_{i-1} - \text{size}_{i-1}| \\
 &= 3
 \end{aligned}$$

Case 2. $\frac{\text{num}_{i-1}-1}{\text{size}_{i-1}} < \frac{1}{3}$ (a contraction is triggered in the i th operation).

Then we have $c_i = \text{num}_i + 1 = \text{num}_{i-1}$ and $\text{size}_i = \frac{2}{3} \cdot \text{size}_{i-1}$.

$$\begin{aligned}
 \hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
 &= c_i + |2 \cdot \text{num}_i - \text{size}_i| - |2 \cdot \text{num}_{i-1} - \text{size}_{i-1}| \\
 &= \text{num}_{i-1} + |2 \cdot (\text{num}_{i-1} - 1) - \frac{2}{3} \cdot \text{size}_{i-1}| - |2 \cdot \text{num}_{i-1} - \text{size}_{i-1}| \\
 &= \text{num}_{i-1} + |2 \cdot \text{num}_{i-1} - \frac{2}{3} \cdot \text{size}_{i-1} - 2| - |2 \cdot \text{num}_{i-1} - \text{size}_{i-1}|
 \end{aligned}$$

Lemma 5.

$$2 \cdot \text{num}_{i-1} - \frac{2}{3} \cdot \text{size}_{i-1} - 2 < 0$$

Proof.

$$\begin{aligned} \frac{\text{num}_{i-1} - 1}{\text{size}_{i-1}} < \frac{1}{3} &\implies 3 \cdot (\text{num}_{i-1} - 1) < \text{size}_{i-1} \\ &\implies 2 \cdot (\text{num}_{i-1} - 1) < \frac{2}{3} \cdot \text{size}_{i-1} \\ &\implies 2 \cdot \text{num}_{i-1} - \frac{2}{3} \cdot \text{size}_{i-1} - 2 < 0 \end{aligned}$$

□

Lemma 6.

$$\forall \text{num}_{i-1} \geq 3, 2 \cdot \text{num}_{i-1} - \text{size}_{i-1} < 0$$

Proof.

$$\begin{aligned} \frac{\text{num}_{i-1} - 1}{\text{size}_{i-1}} < \frac{1}{3} &\implies 3 \cdot (\text{num}_{i-1} - 1) < \text{size}_{i-1} \\ &\implies 2 \cdot \text{num}_{i-1} - 3 + \text{num}_{i-1} < \text{size}_{i-1} \\ &\implies 2 \cdot \text{num}_{i-1} - \text{size}_{i-1} + (\text{num}_{i-1} - 3) < 0 \\ &\implies 2 \cdot \text{num}_{i-1} - \text{size}_{i-1} < 3 - \text{num}_{i-1} \end{aligned}$$

Clearly, $\forall \text{num}_{i-1} \geq 3, 3 - \text{num}_{i-1} \leq 0$.

□

The subcase of $\text{num}_{i-1} < 3$ is trivial. Assume $\text{num}_{i-1} \geq 3$ in the later analysis.

Then

$$\begin{aligned} \hat{c}_i &= \text{num}_{i-1} + |2 \cdot \text{num}_{i-1} - \frac{2}{3} \cdot \text{size}_{i-1} - 2| - |2 \cdot \text{num}_{i-1} - \text{size}_{i-1}| \\ &= \text{num}_{i-1} - (2 \cdot \text{num}_{i-1} - \frac{2}{3} \cdot \text{size}_{i-1} - 2) \overset{(-)}{+} (2 \cdot \text{num}_{i-1} - \text{size}_{i-1}) \\ &= (\text{num}_{i-1} - \frac{1}{3} \cdot \text{size}_{i-1} - 1) + 3 \end{aligned}$$

Lemma 7.

$$\text{num}_{i-1} - \frac{1}{3} \cdot \text{size}_{i-1} - 1 < 0$$

Proof.

$$\begin{aligned} \frac{\text{num}_{i-1} - 1}{\text{size}_{i-1}} < \frac{1}{3} &\implies \text{num}_{i-1} - 1 < \frac{1}{3} \cdot \text{size}_{i-1} \\ &\implies \text{num}_{i-1} - \frac{1}{3} \cdot \text{size}_{i-1} - 1 < 0 \end{aligned}$$

□

Hence

$$\hat{c}_i = (\text{num}_{i-1} - \frac{1}{3} \cdot \text{size}_{i-1} - 1) + 3 < 3$$

Updating...