Dynamic Tables

→ Why?

→ $\alpha(T)$, load factor

→ Array of slots

→ Table expansion

→ Twice as many slots as the previous one

→ load factor $\geq 1/2$

→ create new table, move all elements,

free old table

→ n Table-insert operations on an initially empty table

$c_i \in O(n)$ in worst case

→ $O(n^2)$ total

$$c_i = \begin{cases} i, & \text{if } i-1 = 2^k \\ 1, & \text{otherwise} \end{cases}$$

$$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\log n} 2^j$$

$$< n + 2n$$

$$= 3n$$

Why $3n$: 1 for insertion, 1 for moving,

1 for moving another item.

Potential method

$\phi(T) = 2 \cdot T.num - T.size$

Immediately after an expansion $T.num = T.size/2$

$\phi(T) = 0$

Immediately before an expansion

$\phi(T) = T.num.$

$T.num \geq \frac{T.size}{2}$

$\therefore \phi(T) \geq 0$

$num_0 = 0, \quad size_0 = 0, \quad \phi_0 = 0$

q_i does not trigger expansion

$\hat{c}_i = c_i + \phi_i - \phi_{i-1}$

$= 1 + (2 \cdot num_i - size_i) - (2\, num_{i-1} - size_{i-1})$

$2(num_i - 1)$

$= 3.$

does trigger expansion $\quad size_i = 2\, size_{i-1}$

$\hat{c}_i = c_i + \phi_i - \phi_{i-1} \qquad size_{i-1} = num_{i-1}$

$= num_i - 1$

$= num_i + (2\, num_i - size_i)$

$- (2\, num_{i-1} - size_{i-1}) \qquad size_i = 2(num_i - 1)$

$= 3$

Table expansion and contraction

→ why does simple halving & doubling not work?

$$\phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{if } \alpha(T) \geq 1/2 \\ \frac{T.size}{2} - T.num & \text{if } \alpha(T) < 1/2 \end{cases}$$

implies $\Phi(T) = T.num$, and thus the potential can pay for a contraction if an item is deleted.

To analyze a sequence of n TABLE-INSERT and TABLE-DELETE operations, we let c_i denote the actual cost of the ith operation, $\widehat{c}_i$ denote its amortized cost with respect to Φ, num_i denote the number of items stored in the table after the ith operation, $size_i$ denote the total size of the table after the ith operation, α_i denote the load factor of the table after the ith operation, and Φ_i denote the potential after the ith operation. Initially, $num_0 = 0$, $size_0 = 0$, $\alpha_0 = 1$, and $\Phi_0 = 0$.

We start with the case in which the ith operation is TABLE-INSERT. The analysis is identical to that for table expansion in Section 17.4.1 if $\alpha_{i-1} \geq 1/2$. Whether the table expands or not, the amortized cost $\widehat{c}_i$ of the operation is at most 3. If $\alpha_{i-1} < 1/2$, the table cannot expand as a result of the operation, since the table expands only when $\alpha_{i-1} = 1$. If $\alpha_i < 1/2$ as well, then the amortized cost of the ith operation is

$$\begin{aligned}
\widehat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) \\
&= 1 + (size_i/2 - num_i) - (size_i/2 - (num_i - 1)) \\
&= 0\,.
\end{aligned}$$

If $\alpha_{i-1} < 1/2$ but $\alpha_i \geq 1/2$, then

$$\begin{aligned}
\widehat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + (2 \cdot num_i - size_i) - (size_{i-1}/2 - num_{i-1}) \\
&= 1 + (2(num_{i-1} + 1) - size_{i-1}) - (size_{i-1}/2 - num_{i-1}) \\
&= 3 \cdot num_{i-1} - \frac{3}{2} size_{i-1} + 3 \\
&= 3\alpha_{i-1} size_{i-1} - \frac{3}{2} size_{i-1} + 3 \\
&< \frac{3}{2} size_{i-1} - \frac{3}{2} size_{i-1} + 3 \\
&= 3\,.
\end{aligned}$$

Thus, the amortized cost of a TABLE-INSERT operation is at most 3.

We now turn to the case in which the ith operation is TABLE-DELETE. In this case, $num_i = num_{i-1} - 1$. If $\alpha_{i-1} < 1/2$, then we must consider whether the operation causes the table to contract. If it does not, then $size_i = size_{i-1}$ and the amortized cost of the operation is

$$\begin{aligned}
\widehat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + (size_i/2 - num_i) - (size_{i-1}/2 - num_{i-1}) \\
&= 1 + (size_i/2 - num_i) - (size_i/2 - (num_i + 1))
\end{aligned}$$

If $\alpha_{i-1} < 1/2$ and the ith operation does trigger a contraction, then the actual cost of the operation is $c_i = num_i + 1$, since we delete one item and move num_i items. We have $size_i/2 = size_{i-1}/4 = num_{i-1} = num_i + 1$, and the amortized cost of the operation is

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

When the ith operation is a TABLE-DELETE and $\alpha_{i-1} \geq 1/2$, the amortized cost is also bounded above by a constant. We leave the analysis as Exercise 17.4-2.

In summary, since the amortized cost of each operation is bounded above by a constant, the actual time for any sequence of n operations on a dynamic table is $O(n)$.

Exercises

17.4-1
Suppose that we wish to implement a dynamic, open-address hash table. Why might we consider the table to be full when its load factor reaches some value α that is strictly less than 1? Describe briefly how to make insertion into a dynamic, open-address hash table run in such a way that the expected value of the amortized cost per insertion is $O(1)$. Why is the expected value of the actual cost per insertion not necessarily $O(1)$ for all insertions?

17.4-2
Show that if $\alpha_{i-1} \geq 1/2$ and the ith operation on a dynamic table is TABLE-DELETE, then the amortized cost of the operation with respect to the potential function (17.6) is bounded above by a constant.

17.4-3
Suppose that instead of contracting a table by halving its size when its load factor drops below $1/4$, we contract it by multiplying its size by $2/3$ when its load factor drops below $1/3$. Using the potential function

$$\Phi(T) = |2 \cdot T.num - T.size| ,$$

show that the amortized cost of a TABLE-DELETE that uses this strategy is bounded above by a constant.