Indian Institute of Information Technology
Design and Manufacturing, Kancheepuram
Chennai 600 127, India
An Autonomous Institute under MHRD, Govt of India
http://www.iiitdm.ac.in
COM 501 Advanced Data Structures and Algorithms

Instructor
N.Sadagopan
Scribe:
Kuladeep
Renjith.P

Amortized Analysis - Part 2 - Dynamic Tables

Objective: In this lecture, we shall explore Dynamic tables and its amortized analysis in detail.

Dynamic Tables

Programming languages such as C++ and Java supports 'vector' which is a dynamic array (table) that grows or shrinks based on the context (application). In this section, we shall analyze dynamic tables in detail by considering two operations, namely, insert and delete. Since an application may perform a sequence of insert and delete operations in some order, it is appropriate to investigate dynamic tables from the perspective of amortized analysis.
It is important to come up with strategies for 'growing the table' and 'shrinking the table' so that the underlying application handles requests (insert/delete) nicely and the application does not go for 'expansion' or 'contraction' frequently.

Whenever the table overflows as the current table size can not handle new requests, we grow the table by allocating a new, larger table and free the space of the old table. Similarly if many objects are deleted from the table, it may be worthwhile to reallocate the table with a smaller size. Towards this end, we adopt the following strategy;
The size of the table doubles as the table gets filled. i.e., the size expands from $2^x \longrightarrow 2^{x+1}$. Similarly, we shall discuss later a strategy for contraction if too many deletions happen in a row.

Insertion in a Dynamic Table

Consider a sequence of n insertions. The worst-case time to execute one insertion is $\Theta(n)$. This is true because i^{th} insertion triggers expansion. i.e., for i^{th} insertion there is no free slot in the table and the table must be expanded to accommodate the new element. Time for copying the contents of old table to the new table is $\theta(n)$ followed by insertion which incurs $\theta(1)$. Therefore, the worst-case time for n insertions is $n \cdot \Theta(n) = \Theta(n^2)$. But this bound is not tight because, expansion does not occur so often in the course of n operations. Using amortized analysis, we see that the worst-case cost for n insertions is only $\Theta(n)$. We assume that $n = 2^k$, where n is the size of the table.

Aggregate Analysis

Initially the table is empty. i^{th} operation triggers an expansion only when $i-1$ is a power of 2. $2^x \longrightarrow 2^{x+1}$ (expansion cost is 2^x). Therefore, the total cost of n insert operations is,

$$AC = \underbrace{n}_{1 \text{ per each insert}} + \underbrace{\sum_{i=1}^{\lfloor \log n \rfloor} 2^i}_{\text{expansion cost} < 2n}$$

$$< 3n \qquad = O(n)$$

Amortized Cost $= \frac{O(n)}{n} = O(1)$. Therefore, the cost of each insert is $O(1)$ amortized.

Analysis using a potential function

Amortized cost= Actual cost + change in potential.
We shall work with the following potential function;
$\phi(T) = 2\ Num(T) - Size(T)$.
$N = Num(T)$, the number of elements in table T. $S = Size(T)$, the size of the table T.
Initially, $Num(T) = 0$ and $Size(T) = 0$. Therefore, $\phi(T) = 0$
Note: potential function must be defined in such a way that the potential associated with data structure must be positive for all configurations. i.e., $\phi(T) \geq 0$ for all configurations.
Remark:
1. If $Num(T) = Size(T) \Longrightarrow \phi(T) = Num(T)$
2. If $Num(T) = \frac{Size(T)}{2} \Longrightarrow \phi(T) = 0$

3. **Load Factor** $\alpha = \frac{Num(T)}{Size(T)}$. A measure of how much percentage of table is filled.

Amortized cost for i^{th} Insertion into the Table
$AC_i = c_i + (\phi_i - \phi_{i-1})$
There are two cases possible here, either the i^{th} insertion triggers an expansion or it does not trigger an expansion.
Case 1: i^{th} insertion does not trigger an expansion.
$S_{i-1} = S_i \qquad N_i = N_{i-1} + 1$
$AC_i = 1 + 2N_i - S_i - 2(N_i - 1) + S_{i-1}$
$= 1 + 2N_i - S_i - 2N_i + 2 + S_i$
$= 3$

Case 2: i^{th} insertion triggers an expansion. Since contents of the old table must be copied to the new table followed by insert, the actual cost is N_i.
$S_{i-1} = \frac{S_i}{2} \qquad N_{i-1} = N_i - 1$
$AC_i = N_i + 2N_i - S_i - 2(N_i - 1) + S_{i-1}$
$= N_i + 2N_i - S_i - 2N_i + 2 + \frac{S_i}{2}$
$= N_i + 2 - \frac{S_i}{2}$
But, $N_i = \frac{S_i}{2} + 1$
$AC_i = \frac{S_i}{2} + 1 + 2 - \frac{S_i}{2} = 3$

Thus, the amortized cost of insert is $3 = O(1)$. Therefore, for a sequence of n inserts, the amortized cost is $n \cdot O(1) = O(n)$.

Remark:

1. Once the table expands, the potential associated with it is zero and when the table is full, the potential is $Num(T)$. In all other configurations, the potential is between 0 and $Num(T)$ and thus, $\phi(T) \geq 0$ always.

Amortized cost for Deletion

$AC_i = c_i + \phi_i - \phi_{i-1}$
$= 1 + 2N_i - S_i - (2N_{i-1} - S_{i-1})$
$= 1 + 2(N_{i-1} - 1) - S_i - 2N_{i-1} + S_i \qquad$ Since $S_i = S_{i-1},\ N_i = N_{i-1} - 1$
$= -1$

Thus, the amortized cost of delete is $-1 = O(1)$.
Remark: In our earlier discussion, we focused on expansion when a new insert triggers expansion. However, we did not focus on contraction when the table has a few elements. In the next section, we shall discuss a strategy for contraction.

Strategy for expansion and contraction

Strategy for expansion is same as the strategy we discussed before. Expand when the table is full and create a new table whose size is twice the size of the old table. The strategy for contraction; if the table is half full and i^{th} operation which is delete makes the table size go below $\frac{size}{2}$, then perform contraction. Although, these strategies appear good, it triggers too many expansions and contractions as we can see from the following example.

Consider the following sequence of operations on the Table:
$\underbrace{I, I, I, I, I, ...}_{\frac{n}{2}\ inserts}, \underbrace{I, D, D, I, I, D, D, I, I, D, D, I, ...}_{\frac{n}{2}\ operations}$

$\frac{n}{2}+1$ operation is insert, which triggers an expansion. Subsequently, the table will contain $\frac{size}{2}+1$ elements. The subsequent two deletions brings the table size to $\frac{size}{2}-1$ which inturn triggers a contraction. For the above example, we can see that expansion and contraction happen alternately for $\frac{n}{4}$ times. Let us analyze the cost for the above sequence;
The first $\frac{n}{2}$ operations incurs an amortized cost of $3 \cdot O(n) = O(n)$.
The second $\frac{n}{2}$ operations incurs an amortized cost of $O(n) \cdot (n/2) = O(n^2)$.
The total cost of n operations is $O(n^2)$ and the amortized cost of each operation is $O(n)$.
Drawback of this strategy:
-After an expansion, we do not perform enough deletions to pay for a contraction.
-After a contraction, we do not perform enough insertions to pay for an expansion.

Modified Strategy: We can improve this strategy by allowing $\alpha(T)$ to drop below $\frac{1}{2}$. We continue to double the size when an object is inserted into a full table but, we contract the table when deletion causes $\alpha < \frac{1}{4}$. Therefore, $\frac{1}{4} \leq \alpha(T) \leq 1$.

We now define a new Potential function corresponding to the new strategy,

$\phi(T) = 2Num(T) - Size(T) \qquad for \quad \alpha \geq \frac{1}{2}$

$\quad = \frac{Size(T)}{2} - Num(T) \qquad for \quad \alpha < \frac{1}{2}$

Now, load factor α oscillates between $\frac{1}{4}$ and 1. Most importantly, the choice of potential function from the above two is also dictated by α. Suppose, i^{th} operation is an insert, the possible conditions are:

Case 1: $\alpha_{i-1} \geq \frac{1}{2} \rightarrow \alpha_i \geq \frac{1}{2}$:

Analysis is identical to that of table expansion in previous section $AC_i = 3$

Case 2: $\alpha_{i-1} < \frac{1}{2} \rightarrow \alpha_i < \frac{1}{2}$:

$AC_i = c_i + \phi_i - \phi_{i-1}$
$\quad = 1 + (\frac{S_i}{2} - N_i) - (\frac{S_{i-1}}{2} - N_{i-1})$
$\quad = 1 + \frac{S_i}{2} - N_i - \frac{S_{i-1}}{2} + N_i - 1$
$\quad = 0$
this operation comes for free, no need to spend anything from potential.

Case 3: $\alpha_{i-1} < \frac{1}{2} \rightarrow \alpha_i \geq \frac{1}{2}$:

$$\begin{aligned}
AC_i &= c_i + \phi_i - \phi_{i-1} \\
&= 1 + 2N_i - S_i - (\tfrac{S_{i-1}}{2} - N_{i-1}) \\
&= 1 + 2N_{i-1} + 2 - S_i - \tfrac{S_{i-1}}{2} + N_{i-1}) \\
&= 3 + 3N_{i-1} - \tfrac{3S_{i-1}}{2} \\
&< 3 + \tfrac{3S_{i-1}}{2} - \tfrac{3S_{i-1}}{2} \\
&< 3
\end{aligned}$$

If we charge 1 or 2 per operation, it works fine.

Suppose, i^{th} operation is delete, the possible conditions are:

Case 4: $\alpha_{i-1} \geq \frac{1}{2} \rightarrow \alpha_i \geq \frac{1}{2}$:

$$\begin{aligned}
AC_i &= c_i + \phi_i - \phi_{i-1} \\
&= 1 + 2N_i - S_i - (2N_{i-1} - S_{i-1}) \\
&= 1 + 2N_i - S_i - 2N_i - 2 + S_i \\
&= -1
\end{aligned}$$

Case 5: $\alpha_{i-1} \geq \frac{1}{2} \rightarrow \alpha_i < \frac{1}{2}$:

$$\begin{aligned}
AC_i &= c_i + \phi_i - \phi_{i-1} \\
&= 1 + (\tfrac{S_i}{2} - N_i) - (2N_{i-1} - S_{i-1}) \\
&= 1 + \tfrac{S_i}{2} - N_{i-1} + 1 - 2N_{i-1} + S_i \\
&= 2 + \tfrac{3S_i}{2} - 3N_{i-1} \\
&= 2 + \tfrac{3S_i}{2} - 3N_i - 3 \\
&= -1 \quad \text{Since } \tfrac{N_i}{S_i} < \tfrac{1}{2}, N_i < \tfrac{S_i}{2}.
\end{aligned}$$

$\alpha_{i-1} = \frac{1}{4}$: This triggers a contraction

$S_i = \frac{S_{i-1}}{2}, \quad N_i = \frac{S_{i-1}}{2} \quad and \quad c_i = N_i + 1$

$$\begin{aligned}
AC_i &= c_i + \phi_i - \phi_{i-1} \\
&= 1 + N_i + (\tfrac{S_i}{2} - N_i) - (\tfrac{S_{i-1}}{2} - N_{i-1}) \\
&= 1 + N_i + \tfrac{S_i}{2} - N_i - S_i + \tfrac{S_i}{2} \\
&= 1
\end{aligned}$$

Case 6: $\alpha_{i-1} \leq \frac{1}{2} \rightarrow \frac{1}{4} \leq \alpha_i < \frac{1}{2}$:

$$\begin{aligned}
AC_i &= c_i + \phi_i - \phi_{i-1} \\
&= 1 + \tfrac{S_i}{2} - N_i - \tfrac{S_{i-1}}{2} + N_{i-1}
\end{aligned}$$

Rearrange the terms;

$$\begin{aligned}
&= 1 + N_{i-1} + \tfrac{S_i}{2} - N_{i-1} + 1 - \tfrac{S_i}{2} \\
&= 2
\end{aligned}$$

Therefore, the amortized cost of deletion is at most 2, which is $O(1)$.

Since the amortized cost of each operation is bounded above by a constant, for any sequence of n operations consisting of insert and delete, on a Dynamic Table is $O(n)$ and $O(1)$ amortized per operation.

Acknowledgements: Lecture contents presented in this module and subsequent modules are based on the following text books and most importantly, author has greatly learnt from lectures by algorithm exponents affiliated to IIT Madras/IMSc; Prof C. Pandu Rangan, Prof N.S.Narayanaswamy, Prof Venkatesh Raman, and Prof Anurag Mittal. Author sincerely acknowledges all of them. Special thanks to Teaching Assistants Mr.Renjith.P and Ms.Dhanalakshmi.S for their sincere and dedicated effort and making this scribe possible. Author has benefited a lot by teaching this course to senior undergraduate students and junior undergraduate students who have also contributed to this scribe in many ways. Author sincerely thanks all of them.

References:

1. E.Horowitz, S.Sahni, S.Rajasekaran, Fundamentals of Computer Algorithms, Galgotia Publications.
2. T.H. Cormen, C.E. Leiserson, R.L.Rivest, C.Stein, Introduction to Algorithms, PHI.
3. Sara Baase, A.V.Gelder, Computer Algorithms, Pearson.
4. Eva Tardos and Kleinberg, Algorithm Design, Pearson.