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.

1 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 cannot 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.

1.1 Insertion in a Dynamic Table

Consider a sequence $(I_1, \ldots, I_n)$ of n insertions. If i^{th} insert does not trigger an expansion, i.e., there is a free slot in the table, then the cost of insert is $O(1)$. Otherwise, the table is full. In such a case, the cost of insert includes the cost of expanding the table to a larger size.
Initially, the table size is one, the cost of I_1 is one. For I_2, since the table is full, we create a new table whose size is twice the size of the previous size. That is, we create a table of size two and insert I_2. The cost of I_2 is $1 + 1 = 2$ (1: for copy, 1: for insert). For I_3, the table is full again, we create a new table of size 4 and insert I_3. The cost of I_3 is $2 + 1 = 3$. For I_4, it is just one as there is a free slot. For I_5, it is $4 + 1 = 5$, and so on. In general, if i^{th} insert triggers expansion, then the cost is $2^{i-1} + 1$. We next present the aggregate analysis by considering a sequence of n insertions.

Aggregate Analysis

A trivial analysis shows that the worst-case time to execute one insertion is $\Theta(n)$. This is true because i^{th} insertion triggers an 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 $n \cdot \Theta(n) = \Theta(n^2)$, and hence the cost of insert is $\theta(n)$ amortized. But this bound is not tight because, the expansion does not occur so often in the course of n operations.
Initially the table is empty. i^{th} operation triggers an expansion only when $i - 1$ is a power of 2. The size of the table is increased from 2^{i-1} to 2^i (expansion cost is 2^{i-1}). 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;
$\phi(T) = 2 \cdot Num(T) - Size(T)$.
$N = Num(T)$, 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.
Remarks:
1. If $Num(T) = Size(T) \implies \phi(T) = Num(T)$
2. If $Num(T) = \frac{Size(T)}{2} \implies \phi(T) = 0$

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 $n \cdot O(1) = O(n)$.

Some Observations:

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.

2. This potential function ensures that when the table is full, the table has sufficient potential to supply charges for expansion, and just after expansion of the table, the potential is zero.

3. One can also work with the potential function $\phi(T) = 3 \cdot Num(T) - Size(T)$.

Cost of Insert through Accounting Method:
As part of accounting method, we charge '3' credits for each insert irrespective of whether the insert triggers an expansion or not. This implies that the amortized cost of insert is $3 = O(1)$. We shall now justify the significance of '3' for insert operation; out of 3 credits, 1 credit is used for insert and the other '2' credits are stored with the table. Interestingly, when the table is full and due for expansion, the credits stored at the table is at least the number of elements, and hence, the cost for expansion is supplied by the credits stored at the table.
When a table expands from 2^{i-1} to 2^i, the cost of expansion is supplied by the table itself and just after expansion, there are no credits with the table. We ensure that we perform sufficient inserts before we go for the next expansion. The next set of 2^i requests with '3' credits each ensure that $2 \cdot 2^i$ credits are stored with the table itself. Thus, when we expand from 2^i to 2^{i+1}, the table has $2 \cdot 2^i = 2^{i+1}$ credits which takes care of the expansion cost. The extra credits, $2 \cdot 2^i$ stored at the table can also be seen this way; the input sequence $I = (I_1, \ldots, I_{2^i}, I_{2^i+1}, I_{2^i+2}, \ldots, I_{2^{i+1}})$ is such that, out of 3 credits charged for I_{2^i+1}, one credit is used for the actual insert, the second credit is stored with the element itself, and the third credit is given to I_1. Similarly, three credits supplied to I_{2^i+2} is such that one credit is for the actual cost, the second credit is stored at the element itself, and the third credit is given to I_2. This procedure ensures that each element in the table has one credit each before the next expansion takes place. Note that one can also charge credits '4' or some constant $k \geq 3$ for each insert as part of accounting method.

1.2 Amortized cost: A sequence with Insertion and Deletion

- **Potential Function Method:** Let us analyse the cost of a sequence of insert and delete using the potential function defined in the previous section. The cost of insert is same as before. The cost of delete is,
  $AC_i = c_i + \phi_i - \phi_{i-1}$

$$\begin{aligned} &= 1 + 2N_i - S_i - (2N_{i-1} - S_{i-1}) \\ &= 1 + 2(N_{i-1} - 1) - S_i - 2N_{i-1} + S_i \qquad \text{Since } S_i = S_{i-1},\ N_i = N_{i-1} - 1 \\ &= -1 \end{aligned}$$

  The natural questions are; what does the above number -1 signify and does this potential function is positive for all configurations. It turns out that this potential function is not the appropriate function for deletion as it is not positive in all configurations. For example, if $N_i = 3$ and $S_i = 8$, then $2N_i - S_i = -2$ and thus not the right function to work with. However, this function is good as long as the load factor $\alpha = \frac{1}{2}$ on deletion. Is there a potential function that works fine for all configurations? How about $\phi = 3N_i - S_i$. Interestingly, this function works fine as long as $\alpha = \frac{1}{3}$ on deletion. For example, if $N_i = 5$ and $S_i = 16$, then $3N_i - S_i = -1$ and this function fails. Similar reasoning holds good even if $\phi = c \cdot N_i - S_i$. This suggests us to look for a different scheme to handle deletion operations. Observe that when $\alpha = \frac{1}{2}$ on deletion, the table is half-empty and it is natural to think of contracting the table by half and work with the half of the original size. This reduction in size ensures that potential is positive for subsequent deletions until $\alpha = \frac{1}{2}$ with respect to the new configuration.

  Consider a table with $S_i = 16$, $N_i = 9$, on deletion, $\alpha = \frac{1}{2}$. We now reduce the table size by half; $S_{i+1} = 8, N_{i+1} = 8$. Clearly, the potential function $2N_i - S_i$ is positive for all configurations. It is positive for the next four deletions, after which, $S_{i+5} = 8, N_{i+5} = 4$ and thus we go for contraction to reduce the table size by half further. Let us analyze the cost of deletion for this modified strategy.

  `Deletion with no contraction:`
  $AC_i = c_i + \phi_i - \phi_{i-1}$

$= 1 + 2N_i - S_i - (2N_{i-1} - S_{i-1})$
$= 1 + 2(N_{i-1} - 1) - S_i - 2N_{i-1} + S_i$ Since $S_i = S_{i-1},\ N_i = N_{i-1} - 1$
$= -1$

`Deletion with contraction:`
On deletion, if $\alpha_i = \frac{1}{2}$, then perform contraction. $S_i = \frac{S_{i-1}}{2}$. The actual cost is $1 + N_i$; '1' for the cost of deletion and N_i, the cost of copying to the new table. Note that $N_i = N_{i-1} - 1$
$AC_i = c_i + \phi_i - \phi_{i-1}$

$= 1 + N_i + 2N_i - S_i - (2N_{i-1} - S_{i-1})$
$= 1 + N_i + 2N_i - S_i - 2N_i - 2 + 2S_i$
$= -1 + N_i + S_i$, Since $N_i = S_i$
$= -1 + 2N_i$
$= O(n)$ amortized. Although, this potential function with a scheme for contraction works fine, it does not yield a constant amortized cost for a deletion operation. In the next section, we shall present a different strategy for deletion with contraction which yield a constant amortized cost.

- **Accounting Method:** As part of accounting method, we charge '3' credits for insert and $O(n), n = N_i$ credits for deletion. To be precise, we charge $2N_i$ credits, out of which N_i credits are used for copying into a smaller table in case of contraction, and the remaining N_i credits are stored with the elements itself. The stored credit supplies the charge for subsequent deletions, and hence the subsequent deletions are free as long as the load factor is more than half.

- **Aggregate Analysis:** Consider a sequence of n operations with l insert and the rest are deletion operations. The average cost in worst case is given by
$\frac{l \cdot O(?) + (n-l) \cdot O(?)}{n}$
The worst case scenario happens when the first $\frac{n}{2} + 1$ are insert, followed by alternate delete and insert for $\frac{n}{2} - 1$ times. Assume that $n = 2^k$ for some k, then $\frac{n}{2} + 1$ insert triggers an expansion incurring $O(n)$ cost, followed by the deletion which triggers a contraction, incurring $O(n)$ cost. Thus, the cost of insert is $O(n)$ for $\frac{n}{2} + 1$ inserts and the subsequent insert and delete incur $O(n)$ each. Thus, the amortized cost is $\frac{O(n) + O(n) \cdot O(n)}{n} = O(n)$

Although, the strategy presented for insertion and deletion work fine, it does not yield a constant amortized cost for insertion and deletion.

1.3 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. In the previous section, we performed contraction when, on deletion, α becomes $\frac{size}{2}$. Here, we allow one more deletion and perform contraction when α goes below $\frac{size}{2}$. Although, this strategy 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 $O(n)$.
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 $O(n)$.
This strategy too yield $O(n)$ amortized for insert and delete.

Drawbacks 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.
- After an expansion, we perform just 2 deletions ($O(1)$ operations) before we go for a contraction which incurs $O(n)$ effort.
- How about performing enough deletions, $O(n)$ deletions before the next contraction, instead of just $O(1)$ deletions. Moreover, this increases the number of operations which inturn reduces the average cost in worst case.
- Instead of just 2 deletions, perform either $\frac{n}{4}$ or some function n deletions before go for a contraction.
- For example, after an expansion, suppose we perform $O(n)$ deletions before we go for a contraction or after a contraction we perform $O(n)$ insertions before we go for an expansion, and assume such a sequence appear y times. Then, the amortized cost is $\frac{y \cdot O(n)}{y \cdot O(n)} = O(1)$.

Modified Strategy: 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, $\frac{1}{4} \leq \alpha \leq 1$.

We now define a new potential function corresponding to the new strategy. Note that $\phi = 2N_i - S_i$ does not work fine when $\alpha < \frac{1}{2}$ and hence we need a different potential function if α is less than $\frac{1}{2}$.

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

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

Now, the load factor α oscillates between $\frac{1}{4}$ and 1. Most importantly, the choice of potential function described above is also dictated by α. Observe that,

Just before expansion; $\alpha = 1$, $\phi = 2N_i - S_i = 2N_i - N_i = N_i$
Just after expansion; $\alpha = \frac{1}{2}$, $\phi = 2N_i - S_i = 2\frac{S_i}{2} - S_i = 0$

That is, just after expansion, the potential associated with the table is zero, and each insert increases the potential by 2, when the table is full, the potential is N_i which is sufficient enough to pay for an expansion. Similarly,

Just before contraction; $\alpha = \frac{1}{4}, \phi = \frac{S_i}{2} - N_i = \frac{S_i}{2} - \frac{S_i}{4} = \frac{S_i}{4} = N_i$
Just after contraction; $\alpha = \frac{1}{2}, \phi = 2N_i - S_i = 2\frac{S_i}{2} - S_i = 0$

That is, just before contraction, the table has sufficient potential to supply for the contraction, and just after the contraction, the potential is zero.
We shall now analyze the cost of insert and delete with respect to this new function.

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

Case 1: $\alpha_{i-1} \geq \frac{1}{2}$ and $\alpha_i \geq \frac{1}{2}$
Analysis is identical to that of table expansion discussed in Section 1.1, therefore, amortized cost $AC_i = 3$.

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

$AC_i = c_i + \phi_i - \phi_{i-1}$
$= 1 + (\frac{S_i}{2} - N_i) - (\frac{S_{i-1}}{2} - N_{i-1})$
$= 1 + \frac{S_i}{2} - N_i - \frac{S_{i-1}}{2} + N_i - 1$
$= 0$
this operation comes for free, which means, the table has potential to supply for this operation. We shall justify this through an example after discussing the delete operation.

Case 3: $\alpha_{i-1} < \frac{1}{2}$ $\alpha_i \geq \frac{1}{2}$
$AC_i = c_i + \phi_i - \phi_{i-1}$
$= 1 + 2N_i - S_i - (\frac{S_{i-1}}{2} - N_{i-1})$
$= 1 + 2N_{i-1} + 2 - S_i - \frac{S_{i-1}}{2} + N_{i-1}$
$= 3 + 3N_{i-1} - \frac{3S_{i-1}}{2}$
Since, $\frac{N_{i-1}}{S_{i-1}} < \frac{1}{2}$
$< 3 + \frac{3S_{i-1}}{2} - \frac{3S_{i-1}}{2}$
$< 3.$
Further,

$AC_i = 1 + 2N_i - S_i - (\frac{S_{i-1}}{2} - N_{i-1})$
$= 1 + 2N_i - S_i - \frac{S_i}{2} + N_i - 1$
$= 3N_i - \frac{3}{2}S_i$
Since, $\frac{N_i}{S_i} \geq \frac{1}{2}$
$AC_i \geq \frac{3}{2}S_i - \frac{3}{2}S_i = 0$
This implies that, insert of this type requires some credit between 0 and 2.

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

Case 4: $\alpha_{i-1} \geq \frac{1}{2}$ and $\alpha_i \geq \frac{1}{2}$
$AC_i = c_i + \phi_i - \phi_{i-1}$
$= 1 + 2N_i - S_i - (2N_{i-1} - S_{i-1})$
$= 1 + 2N_i - S_i - 2N_i - 2 + S_i$
$= -1$

Case 5: $\alpha_{i-1} \geq \frac{1}{2}$ and $\alpha_i < \frac{1}{2}$
$AC_i = c_i + \phi_i - \phi_{i-1}$
$= 1 + (\frac{S_i}{2} - N_i) - (2N_{i-1} - S_{i-1})$
$= 1 + \frac{S_i}{2} - N_{i-1} + 1 - 2N_{i-1} + S_i$
$= 2 + \frac{3S_i}{2} - 3N_{i-1}$
$= 2 + \frac{3S_{i-1}}{2} - 3N_{i-1}$
Since $\frac{N_{i-1}}{S_{i-1}} \geq \frac{1}{2}$, $AC_i \leq 2 + \frac{3}{2}S_{i-1} - \frac{3}{2}S_{i-1}$
Thus, $AC_i \leq 2$.
Further, $AC_i = 1 + (\frac{S_i}{2} - N_i) - (2N_{i-1} - S_{i-1})$
$= 1 + \frac{3}{2}S_i - N_i - 2(N_i + 1)$
$= -1 + \frac{3}{2}S_i - \frac{3}{2}N_i$
Since $\frac{N_i}{S_i} < \frac{1}{2}$, we get, $AC_i > -1$ Therefore, we charge some credit between 0 and 2 for delete of this type.

Case 6: Delete with no contraction: $\alpha_{i-1} \leq \frac{1}{2}$ and $\frac{1}{4} \leq \alpha_i < \frac{1}{2}$
$AC_i = c_i + \phi_i - \phi_{i-1}$
$= 1 + \frac{S_i}{2} - N_i - \frac{S_{i-1}}{2} + N_{i-1}$

Rearrange the terms;
$= 1 + N_{i-1} + \frac{S_i}{2} - N_{i-1} + 1 - \frac{S_i}{2}$
$= 2$

Case 7: Delete with contraction: $\alpha_{i-1} = \frac{1}{4}$. This triggers a contraction, the table size reduces by half.

$S_i = \frac{S_{i-1}}{2}$, $N_i = \frac{S_{i-1}}{2}$ and $c_i = N_i + 1$, $N_{i-1} = \frac{S_{i-1}}{4} = \frac{2S_i}{4} = \frac{S_i}{2}$
$AC_i = c_i + \phi_i - \phi_{i-1}$
$= 1 + N_i + (\frac{S_i}{2} - N_i) - (\frac{S_{i-1}}{2} - N_{i-1})$
$= 1 + N_i + \frac{S_i}{2} - N_i - S_i + \frac{S_i}{2}$
$= 1$

In the above discussion, the contraction is performed when α goes below $\frac{1}{4}$. One can also perform contraction, when $\alpha = \frac{1}{4}$ on deletion. 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. Thus, the modified strategy yields the desired result.

Accounting Method: From our discussion on potential function method for dynamic table, it is clear that if we charge '3' credits for insert and '2' credits for delete, any sequence of n operations can be performed keeping the data structure at credit always. However, one can tighten this analysis, i.e., instead of '2' credits for delete always, one can supply credits in the range 0 to 2 depending on the load factor, which we shall discuss now in detail. To answer this, we revisit the potential function analysis and explain the potential associated with each operation through an example case study.

Consider a table of size $S_i = 8$ and $N_i = 8$. For insert, since potential function analysis suggests '3' credits, we charge '3' credits for each insert in the input sequence $(O_1, \ldots, O_8)$. For the next insert, since the table is full, we need to go for an expansion. Note that there are $2N_i$ excess credits available at the table which supply for expansion procedure and hence, there is no credit for expansion. Assume that, we insert 8 more elements with 3 credits each. Now $S_i = 16$ and $N_i = 16$. Assume that the next operation O_{17} is delete.

1. As per potential function method, since $\alpha_i \geq \frac{1}{2}$, the cost of this operation is '-1'. Refer to case 4 of previous section. What does '-1- signify ? It signifies (i) the cost of the current delete is free (ii) take back credit '1' from the table. Note that for O_{16}, the excess credit of '2' is stored at the element itself. Now, when we delete that element, out of '2' credits, one credit is used for performing deletion and the other one credit is taken out of the table. This implies that this operation is free and the element to be deleted itself pays for this operation. Moreover, if we charge uniformly '2' credits for all delete operations, then the excess credit will be stored at the table which may not be used at all by any of the subsequent operations.

2. Assume that the next set of seven operations; O_{18} to O_{24} are delete. Analysis is similar to the above discussion as $\alpha_i \geq \frac{1}{2}$, and hence we charge nothing. Note that the current configuration is $S_i = 16$ and $N_i = 8$. Suppose the next operation is also delete. Then, $\alpha_i < \frac{1}{2}$. Refer to case 5. Now there is no credit associated with the element to be deleted as it was taken away by previous deletions. Thus, we charge 1 credit for this operation. If we charge at least one credit, then the excess credit will be stored at the data structure.

3. Suppose the next operation is insert, then we fall into case 3. We charge at least one credit as there may not be excess credits at the table.

4. Consider the configuration $S_i = 16$ and $N_i = 7$. Assume that the next operation is delete, then it is case 6. We charge '2' credits, of which, one credit is used up for deletion and the other one credit will be stored with the table which will be used later during contraction. For example, suppose the next four operations are also delete, then, $N_i = 3$ and the table will have at least 4 credits which can be used for copying $N_i = 3$ elements to the new table. Case 7 considers the contraction scenario which charges just '1' credit for deleting that element which will bring the load factor below $\frac{1}{4}$ and the cost for contraction is free.

5. Consider the configuration $S_i = 16$ and $N_i = 5$. Suppose the next operation is insert, then it is case 2 and we need not charge '3' credits for this insert. The earlier delete of case 6 has supplied some excess credits to the table which will supply for this insert operation, and hence, this operation is free.

6. This completes our micro-level analysis of dynamic table case study for the modified potential function.

Some interesting observations:
1. Since the load factor α swings between $\frac{1}{4}$ and 1, one can work with $4N_i - S_i$ for all α instead of two different potential functions given in the modified strategy. This function ensures that when $N_i = \frac{S_i}{4}$, the potential is zero and for all other configurations, the potential is positive. The drawback of this potential function is that it does not yield constant amortized for delete with contraction.

2. `Deletion with contraction:` $\phi = 4N_i - S_i$. $AC_i = 1 + N_i + 4N_i - S_i - 4N_{i-1} + S_{i-1}$
Note that $S_i = \frac{S_{i-1}}{2}$, $N_i = N_{i-1} - 1$.
$AC_i = 1 + N_i + 4N_i - \frac{S_{i-1}}{2} - 4N_i - 4 + S_{i-1}$
$= -3 + N_i + \frac{S_{i-1}}{2}$
$= -3 + N_i + \frac{4N_i}{2} = -3 + 3N_i$
$\neq O(1)$. Although, this function is technically correct, it does not yield a constant amortized cost.

3. In general, if α is such that $\frac{1}{k} \leq \alpha \leq 1$ for some fixed k, then $\phi = kN_i - S_i$ works fine. However, it may not yield $O(1)$ amortized cost for all insert and delete operations.

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 thank 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.