Dynamic Tables
Why Dynamic Tables?

In some applications:

• We don't know how many objects will be stored in a table.

• We may allocate space for a table
  – But, later we may find out that it is not enough.
  – Then, the table must be reallocated with a larger size.
    • All the objects stored in the original table
    • Must be copied over into the new table.
Why Dynamic Tables?

• Similarly, if many objects are deleted from the table:
  – it may be worthwhile to reallocate the table with a smaller size.

This problem is called

Dynamically Expanding and Contracting a table.
Why Dynamic Tables?

Using amortized analysis we will show that,

The amortized cost of insertion and deletion is \( O(1) \).

Even though the actual cost of an operation is large when it triggers an expansion or a contraction.

We will also show how to guarantee that

The unused space in a dynamic table never exceeds a constant fraction of the total space.
Operations

**TABLE-INSERT:**

*Inserts* into the table an item that occupies a single slot.

**TABLE-DELETE:**

*Removes* an item from the table & frees its slot.
Load Factor

Load Factor of a Dynamic Table $T$

$$\alpha(T) = \frac{\text{Number of items stored in the table}}{\text{size}(\text{number of slots}) \text{ of the table}}$$

For an empty table

$$\alpha(T) = \frac{0}{0} = 1$$

by definition
Insertion-Only Dynamic Tables

Table-Expansion:

• Assumption:
  – Table is allocated as an array of slots

• A table fills up when
  – all slots have been used
  – equivalently, when its load factor becomes 1

• Table-Expansion occurs when
  – An item is to be inserted into a full table
Insertion-Only Dynamic Tables

• A Common Heuristic
  – Allocate a new table that has twice as many slots as the old one.

• Hence, insertions are performed if only

\[ \frac{1}{2} \leq \alpha(T) \leq 1 \]
Table Insert

TABLE-INSERT (T, x)
    if size[T] = 0 then
        allocate table[T] with 1 slot
        size[T] ← 1
    if num[T] = size[T] then
        allocate new-table with 2.size[T] slots
        copy all items in table[T] into new-table
        free table[T]
        table[T] ← new-table[T]
        size[T] ← 2.size[T]
    insert x into table[T]
    num[T] ← num[T] + 1
end

Initially, table is empty, so num[T] = size[T] = 0
Table Expansion

• Running time of **TABLE-INSERT** is proportional to the number of elementary insertion operations.

• Assign a cost of 1 to each elementary insertion

• Analyze a sequence of \( n \) **TABLE-INSERT** operations on an initially empty table
Cost of Table Expansion

What is cost $c_i$ of the $i$-th operation?

- If there is room in the current table (or this is the first operation)
  
  $c_i = 1$ (only one elementary insert operation)

- If the current table is full, an expansion occurs, then the cost is $c_i = i$.
  
  1 for the elementary insertion of the new item
  
  $i-1$ for the items that must be copied from the old table to the new table.
Cost of Table Expansion

• If \( n \) operations are performed,
  
  The worst case cost of an operation is \( O(n) \)
  
  Therefore the total running time is \( O(n^2) \)

• This bound is not tight, because
  
  Expansion does not occur so often in the course of \( n \) operations.
Amortized Analysis of Insert

The Aggregate Method

Table is *initially empty*.

Observe:

*i*-th operation causes an expansion only when *i*-1 is a power of 2.

\[
c_i = \begin{cases} 
  i & \text{if } i \text{ is an exact power of 2} \\
  1 & \text{otherwise}
\end{cases}
\]
The Aggregate Method

Therefore the total cost of \( n \) TABLE-INSERT operations is

\[
\sum_{i=1}^{n} c_i = n + \sum_{j=0}^{\lfloor \log n \rfloor} 2^j < n + \sum_{j=0}^{\log n} 2^j = n + 2n = 3n
\]

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_i )</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>Expansion cost</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The amortized cost of a single operation is \( 3n/n = 3 = O(1) \)
The Accounting Method

Assign the following amortized costs

- Table-Expansion : 0
- Insertion of a new item : 3

Insertion of a new item

a) 1 (as an actual cost) for inserting itself into the table
b) 1 (as a credit) for moving itself in the next expansion
c) 1 (as a credit) for moving another item (in the next expansion) that has already moved in the last expansion
Accounting Method Example

Size of the table: $M$

Immediately after an expansion (just before the insertion)

\[ \text{num}[T] = \frac{M}{2} \text{ and size}[T] = M \text{ where } M \text{ is a power of 2.} \]

Table contains no credits
Accounting Method Example

1st insertion

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
X & X & X & X & X & X & X & X & Z & X & X & X & X & X & X & X \\
$1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 \\
\end{array}
\]

2nd insertion

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
X & X & X & X & X & X & X & X & Z & Z & X & X & X & X & X & X \\
$1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 & $1 \\
\end{array}
\]

(a) $1 for insertion

(b) $1

(c) $1
Accounting Method Example

\(M/2\)th Insertion

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
</tr>
<tr>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td>$1</td>
<td></td>
</tr>
</tbody>
</table>

Thus, by the time the table contains \(M\) items and is full

– each item in the table has \$1 of credit to pay for its move during the next expansion
The Potential Method

Define a potential function $\Phi$ that is

- 0 immediately after an expansion
- builds to the table size by the time table becomes full

Next expansion can be paid for by the potential.
Definition of $\Phi$

One possible $\Phi$ is:

$$\Phi(T) = 2 \cdot \text{num}[T] - \text{size}[T]$$

Immediately after an expansion

$$\text{size}[T] = 2 \cdot \text{num}[T] \implies \Phi(T) = 0$$

Immediately before an expansion

$$\text{size}[T] = \text{num}[T] \implies \Phi(T) = \text{num}[T]$$

The initial value for the potential is 0
Definition of $\Phi$

Since the table is at least half full (i.e. $\text{num}[T] \geq \text{size}[T] / 2$)

$\Phi(T)$ is always nonnegative.

Thus, the sum of the amortized cost of $n$ TABLE-INSERT operations is an upper bound on the sum of the actual costs.
Analysis of $i$-th Table Insert

$n_i : \text{num}[T]$ after the $i$-th operation

$s_i : \text{size}[T]$ after the $i$-th operation

$\Phi_i : \text{Potential}$ after the $i$-th operation

Initially we have $n_i = s_i = \Phi_i = 0$

Note that, $n_i = n_{i-1} + 1$ always hold.
Insert without Expansion

If the $i$-th TABLE-INSERT does not trigger an expansion,

\[ s_i = s_{i-1} ; c_i = 1 \]

\[ \hat{c}_i = c_i + \phi_i - \phi_{i-1} = 1 + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) \]
\[ = 1 + (2(n_{i-1} + 1) - s_{i-1}) - (2n_{i-1} - s_{i-1}) \]
\[ = 1 + 2n_{i-1} + 2 - s_{i-1} - 2n_{i-1} + s_{i-1} = 3 \]
Insert with Expansion

If the \( i \)-th TABLE-INSERT does trigger an expansion, then

\[
\begin{align*}
    n_{i-1} &= s_{i-1} \\
s_i &= 2s_{i-1} \\
c_i &= n_i = n_{i-1} + 1 \\
\hat{c}_i &= c_i + \phi_i - \phi_{i-1} = n_i + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) \\
&= (n_{i-1} + 1) + (2(n_{i-1} + 1) + 2s_{i-1}) - (2n_{i-1} - s_{i-1}) \\
&= n_{i-1} + 1 + 2n_{i-1} + 2 - 2n_{i-1} - 2n_{i-1} + n_{i-1} = 3
\end{align*}
\]
Adding Delete Operation

TABLE-DELETE: Remove the specified item from the table. It is often desirable to contract the table. In table contraction, we would like to preserve two properties

- **Load factor** of dynamic table is bounded below by a constant
- **Amortized cost** of a table operation is bounded above by a constant

We assume that the cost can be measured in terms of elementary insertions and deletions
Expansion and Contraction

A natural strategy for expansion and contraction

• Double the table size when an item is inserted into a full table
• Halve the size when a deletion would cause \( \alpha(T) < 1 / 2 \)

This strategy guarantees \( \frac{1}{2} \leq \alpha(T) \leq 1 \)

Unfortunately, it can cause the amortized cost of an operation to be quite large
Worst-Case for $\alpha(T) \geq \frac{1}{2}$

Consider the following worst case scenario

- We perform $n$ operations on an empty table where $n$ is a power of 2
- First $n/2$ operations are all insertions, cost a total of $\Theta(n)$
  at the end: we have $\text{num}[T] = \text{size}[T] = n/2$
- Second $n/2$ operations repeat the sequence $\text{I DD I I DD I I DD I ...}$
Worst-Case for \( \alpha(T) \geq \frac{1}{2} \)

Example: \( n=16 \)

<table>
<thead>
<tr>
<th>( i )</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>oper:</td>
<td>I</td>
<td>I</td>
<td>...</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>D</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td>D</td>
<td>D</td>
<td>I</td>
</tr>
<tr>
<td>( n_i )</td>
<td>1</td>
<td>2</td>
<td>...</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>( s_i )</td>
<td>1</td>
<td>2</td>
<td>...</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

In the second \( n/2 \) operations

- The first \text{INSERT} cause an expansion
- Two further \text{DELETEs} cause contraction
- Two further \text{INSERTs} cause expansion ... and so on

Hence there are \( n/8 \) expansions and \( n/8 \) contractions

The cost of each expansion and contraction is \( \approx n/2 \)
Worst-Case for $\alpha(T) \geq \frac{1}{2}$

Thus the total cost of $n$ operations is $\Theta(n^2)$ since
- First $n/2$ operations: $3n$
- Second $n/2$ operations: $(n/4)*(n/2)=n^2/8$

The amortized cost of an operation is $\Theta(n)$

The difficulty with this strategy is
- 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
Improving Expansion – Contraction

We can improve upon this strategy by allowing $\alpha(T)$ to drop below $\frac{1}{2}$

We continue to double the table size when an item is inserted into a full table

But, we halve the table size (perform contraction) when a deletion causes $\alpha(T) < \frac{1}{4}$ rather than $\alpha(T) < \frac{1}{2}$,

Therefore, $\frac{1}{4} \leq \alpha(T) \leq 1$
Improving Expansion – Contraction

Hence after an expansion, $\alpha(T) = \frac{1}{2}$, thus at least half of the items in the table must be deleted before a contraction can occur.

Similarly, after a contraction $\alpha(T) = \frac{1}{2}$, thus the number of items in the table must be doubled by insertions before an expansion can occur.
Potential Method

Define the potential function as follows

• $\Phi(T) = 0$ immediately after an expansion or contraction

• Recall that, $\alpha(T) = \frac{1}{2}$ immediately after and expansion or contraction,
  therefore the potential should build up to $\text{num}[T]$ as $\alpha(T)$ increases to 1 or decreases to $\frac{1}{4}$

• So that the next expansion or contraction can be paid by the potential.
\[ \Phi(\alpha) \text{ w.r.t. } \alpha(T) \]

\[ M = \text{num}[T] \text{ when an expansion or contraction occurs} \]

Diagram showing the relationship between \( \Phi(\alpha) \) and \( \alpha(T) \) with numerical values marked at specific points. The graph illustrates the behavior of the function with respect to the parameter \( \alpha(T) \).
Description of New $\Phi$

One such $\Phi$ is

$$\Phi(T) = \begin{cases} 
2num[T] - size[T] & \text{if } \alpha(T) \geq \frac{1}{2} \\
\frac{size[T]}{2} - num[T] & \text{if } \alpha(T) < \frac{1}{2}
\end{cases}$$

or

$$\Phi(T) = \begin{cases} 
num[T](2 - 1/\alpha) & \text{if } \alpha(T) \geq \frac{1}{2} \\
num[T](1/2\alpha - 1) & \text{if } \alpha(T) < \frac{1}{2}
\end{cases}$$
Description of New $\Phi$

- $\Phi = 0$ when $\alpha = \frac{1}{2}$
- $\Phi = \text{num}[T]$ when $\alpha = \frac{1}{4}$
- $\Phi = 0$ for an empty table
  \[(\text{num}[T] = \text{size}[T]=0, \alpha[T] = 0)\]
- $\Phi$ is always nonnegative
Amortized Analysis

Operations are:

– TABLE-INSERT

– TABLE-DELETE

\[ c_i : \text{Actual Cost} \quad \hat{c}_i : \text{Amortized Cost} \quad \Phi_i : \Phi(T) \]

\[ n_i : \text{num}[T] \quad s_i : \text{size}[T] \quad \alpha_i : \alpha(T) \]

after the \( i \)-th operation
Table Insert

\[ n_i = n_{i-1} + 1 \Rightarrow n_{i-1} = n_i - 1 \]

Table contraction may not occur.

- \( \alpha_{i-1} \geq \frac{1}{2} \)
  Analysis is identical to that for table expansion
  Therefore, \( \hat{c}_i = 3 \) whether the table expands or not.

- \( \alpha_{i-1} < \frac{1}{2} \) and \( \alpha_i < \frac{1}{2} \)
  Expansion may not occur \( (\hat{c}_i = 1, s_i = s_{i-1}) \)
  \[
  \hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + \left(3i/2 - n_i\right) - \left(s_{i-1}/2 - n_{i-1}\right)
  \]
  \[
  = 1 + \frac{s_i}{2} - n_i - \frac{s_i}{2} + (n_i - 1) = 0
  \]
Table Insert

- \( \alpha_{i-1} < \frac{1}{2} \) and \( \alpha_i \geq \frac{1}{2} \)

\( \Rightarrow \alpha_i = \frac{1}{2} \)

Expansion may not occur \((c_i = 1; s_i = s_{i-1}; n_i = s_i / 2)\)

\[
\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (s_i / 2 - n_i) - (2n_{i-1} - s_{i-1}) \\
= 1 + s_i / 2 - n_i - 2(n_i - 1) + s_i = 3 - 3s_i / 2 - 3n_i \\
= 3 + 3s_i / 2 - 3n_i = 3 + 3s_i / 2 - 3s_i / 2 = 3
\]

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

\[ n_i = n_{i-1} - 1 \Rightarrow n_{i-1} = n_i + 1 \]

Table expansion may not occur.

- \( \alpha_{i-1} \leq \frac{1}{2} \) and \( \frac{1}{4} \leq \alpha_i < \frac{1}{2} \) (It does not trigger a contraction)

\[ s_i = s_{i-1} \text{ and } c_i = 1 \text{ and } \alpha_i < \frac{1}{2} \]

\[ \hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (s_i / 2 - n_i) - (s_{i-1} / 2 - n_{i-1}) \]
\[ = 1 + s_i / 2 - n_i - s_i / 2 + (n_i + 1) = 2 \]
Table Delete

- $\alpha_{i-1} = \frac{1}{4}$ (It does trigger a contraction)

\[ s_i = s_{i-1}/2 ; n_i = s_{i-1}/2; \text{ and } c_i = n_i + 1 \]

\[ \hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = (n_i + 1) + (s_i / 2 - n_i) - (s_{i-1} / 2 - n_{i-1}) \]

\[ = n_i + 1 + s_i / 2 - n_i - s_i + s_i / 2 = 1 \]

- $\alpha_{i-1} > \frac{1}{2} (\alpha_i \geq \frac{1}{2})$

Contraction may not occur ($c_i=1 ; s_i = s_{i-1}$)

\[ \hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) \]

\[ = 1 + 2n_i - s_i - 2(n_i + 1) + s_i = -1 \]
• \( \alpha_{i-1} = \frac{1}{2} \) \((\alpha_i < \frac{1}{2})\) 

Contraction may not occur

\[
c_i = 1 \; ; \; s_i = s_{i-1} \; ; \; n_i = s_{i-1}/2; \; \text{and} \; \Phi_{i-1}=0\)
\]

\[
\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (s_i / 2 - n_i) - 0 \\
= 1 + s_i / 2 - n_i \; \; \text{but} \; \; n_{i+1} = s_i / 2 \\
= 1 + (n_i + 1) - n_i = 2
\]
Table Delete

Thus, the \textit{amortized cost} of a \textsc{table-delete} operation is at most 2

Since the \textit{amortized cost} of each operation is \textit{bounded} above by a constant

The \textit{actual time} for any sequence of \textit{n operations} on a Dynamic Table is $O(n)$