Dynamic tables

Problem: Maintenance of a table under the operations \textit{insert} and \textit{delete} such that
1. the table size can be adjusted to the number of elements
2. a fixed portion of the table is always filled with elements
3. the costs for \( n \) insert or delete operations are in \( O(n) \).

Organisation of the table: hash table, heap, stack, etc.

Load factor \( \alpha \): fraction of table spaces of \( T \) which are occupied.

Cost model:
- Insertion or deletion of an element causes cost 1, if the table is not filled yet.
- If the table size is changed, all elements must be copied.

Initialisation

```java
class dynamicTable {
    private int[] table;
    private int size;
    private int num;

    dynamicTable () {
        table = new int[1]; // initialize empty table
        size = 1;
        num = 0;
    }
}
```

Expansion strategy: insert

Double the table size whenever an element is inserted in the fully occupied table!

```java
public void insert (int x) {
    if (num == size) {
        int[] newTable = new int[2*size];
        for (int i=0; i < size; i++)
            insert table[i] in newTable;
        table = newTable;
        size = 2*size;
    }
    insert x in table;
    num = num + 1;
}
```

Insert operation in an initially empty table

\( \xi \) = cost of the \( i \)-th insert operation

Worst case:
- \( \xi = 1 \), if the table was not full before operation \( i \)
- \( \xi = (i-1) + 1 \), if the table was full before operation \( i \)

Hence, \( n \) insert operations require costs of at most

\[ \sum_{i=1}^{n} (i) = O(n^2) \]

Amortized worst case:
- Aggregate analysis, accounting method, potential method

Amortized cost

- Let the real cost of the \( i \)-th insertion be \( \xi \). We want to define the amortized cost \( a_i \), such that

\[ \sum_{i=1}^{n} a_i \geq \sum_{i=1}^{n} \xi = O(n). \]
Potential method

A table with

- \( k = T.\text{num elements} \)
- \( s = T.\text{size spaces} \)

Potential function

\[ \phi(T) = 2k - s \]

We also write \( \phi_i \) for \( \phi(T) \) after the \( i \)-th insert operation.

We now define \( a_i = \phi_i - \phi_{i-1} + t_i \).

Properties of the potential function

- \( \phi_0 = \phi(T_0) = \phi(\text{empty table}) = -1 \)
- For all \( i \geq 1 \): \( \phi_i = \phi(T_i) \geq 0 \)
  - Since \( \phi_n - \phi_0 \geq 0 \), \( \sum a_i \) is an upper bound for \( \sum t_i \)

Amortized cost of insert (1)

Case 1: \([i\text{-th operation does not trigger an expansion}]\)

\[ a_i = \phi_i - \phi_{i-1} + t_i = (2k_i - s_i) - (2k_{i-1} - s_{i-1}) + t_i = 2(\delta_i - \delta_{i-1}) + t_i = 2 - 0 + 1 \leq 3. \]

Amortized cost of insert (2)

Case 2: \([i\text{-th operation triggers an expansion}]\)

\[ a_i = \phi_i - \phi_{i-1} + t_i = (2k_i - s_i) - (2k_{i-1} - s_{i-1}) + t_i = 2(k_i - k_{i-1}) - (s_i - s_{i-1}) + t_i = 2 - s_{i-1} + t_i \leq 3. \]

Insertion and deletion of elements

Now: contract table, if the load is too small!

Goals:

- (1) Load factor is always bounded below by a constant
- (2) Amortized cost of a single insert or delete operation is constant.

First attempt:

- Expansion: same as before
- Contraction: halve the table size as soon as table is less than \( \frac{1}{3} \) occupied (after the deletion)!

"Worst" sequence of insert and delete operations

\[ \sum a_i = \sum (2 - s_{i-1} + t_i) \leq 3n/2 \]

<table>
<thead>
<tr>
<th>Operation</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/2 times insert (table fully occupied)</td>
<td>3n/2</td>
</tr>
<tr>
<td>I, I: expansion</td>
<td>n/2 + 1</td>
</tr>
<tr>
<td>D, D: contraction</td>
<td>n/2 + 1</td>
</tr>
<tr>
<td>I, I: expansion</td>
<td>n/2 + 1</td>
</tr>
<tr>
<td>D, D: contraction</td>
<td>n/2 + 1</td>
</tr>
</tbody>
</table>

Total cost of the sequence

\[ \Omega(n) \]
Second attempt

Expansion: (as before) double the table size, if an element is inserted in the full table.

Contraction: As soon as the load factor is below ¼, halve the table size.

Hence:
At least ¼ of the table is always occupied, i.e. 
\[ \frac{1}{4} \leq \alpha \leq 1 \]

Cost of a sequence of insert and delete operations?

Analysis: insert and delete

\[ \alpha = \frac{k}{s} \]

Potential function

\[ \phi(T) = \begin{cases} 2k - s, & \text{if } \alpha \geq 1/2 \\ s/2 - k, & \text{if } \alpha < 1/2 \end{cases} \]

Homework: Check \( \sum a_i \geq \sum t_i \)

Directly after an expansion or contraction of the table:
\[ s = 2k, \text{ hence } \phi(T) = 0 \]

insert

i-th operation: \( k_i = k_{i-1} + 1 \)

Case 1: \( \alpha_i \geq \frac{1}{2} \)

Case 2: \( \alpha_i < \frac{1}{2} \)

Case 2.1: \( \alpha_i < \frac{1}{2} \) (no expansion)

Case 2.2: \( \alpha_i \geq \frac{1}{2} \) (no expansion)

Case 2.1:
\[ \alpha_{i-1} < \frac{1}{2}, \alpha_i < \frac{1}{2} \]

Potential function

\[ \phi(T) = \begin{cases} 2k - s, & \text{if } \alpha \geq 1/2 \\ s/2 - k, & \text{if } \alpha < 1/2 \end{cases} \]

Potential function

\[ \phi(T) = \begin{cases} 2k - s, & \text{if } \alpha \geq 1/2 \\ s/2 - k, & \text{if } \alpha < 1/2 \end{cases} \]
\[ k_i = k_{i-1} - 1 \]

Case 1: \( \alpha_i < \frac{1}{2} \)

Case 1.1: deletion causes no contraction
\[ s_i = s_{i-1} \]

Potential function \( \phi \)

\[
\phi(T) = \begin{cases} 
2k - s, & \text{if } \alpha \geq \frac{1}{2} \\
\frac{s}{2} - k, & \text{if } \alpha < \frac{1}{2} 
\end{cases}
\]

Case 1.2: \( \alpha_i < \frac{1}{2} \) deletion causes a contraction
\[ 2s_i = s_{i-1} \]

\[ k_{i-1} = \frac{s_{i-1}}{4} \]

Case 2: \( \alpha_i \geq \frac{1}{2} \)

no contraction
\[ s_i = s_{i-1}, \quad k_i = k_{i-1} - 1 \]

Potential function \( \phi \)

\[
\phi(T) = \begin{cases} 
2k - s, & \text{if } \alpha \geq \frac{1}{2} \\
\frac{s}{2} - k, & \text{if } \alpha < \frac{1}{2} 
\end{cases}
\]

Case 2.1: \( \alpha_i \geq \frac{1}{2} \)

\[ s_i = s_{i-1}, \quad k_i = k_{i-1} - 1 \]

Potential function \( \phi \)

\[
\phi(T) = \begin{cases} 
2k - s, & \text{if } \alpha \geq \frac{1}{2} \\
\frac{s}{2} - k, & \text{if } \alpha < \frac{1}{2} 
\end{cases}
\]

Case 2.2: \( \alpha_i < \frac{1}{2} \)