Dynamic Tables
Slides courtesy of Charles Leiserson with small changes by Carola Wenk

Dynamic tables/arrays

**Goal:** Maintain insertions into an array/table, such that the array is as small as possible.

**Problem:** What if we don’t know the number of insertions in advance?

**Solution:** Dynamic tables.

**Idea:** Whenever the table overflows, “grow” it by allocating (via `malloc` or `new`) a new, larger table. Move all items from the old table into the new one, and free the storage for the old table.

Example of a dynamic table

1. INSERT
2. INSERT 
   *overflow*
Example of a dynamic table

1. INSERT
2. INSERT

Example of a dynamic table

1. INSERT
2. INSERT

Example of a dynamic table

1. INSERT
2. INSERT
3. INSERT

overflow
Example of a dynamic table

1. INSERT
2. INSERT
3. INSERT

overflow

Example of a dynamic table

1. INSERT
2. INSERT
3. INSERT

Example of a dynamic table

1. INSERT
2. INSERT
3. INSERT
4. INSERT
Example of a dynamic table

1. INSERT
2. INSERT
3. INSERT
4. INSERT
5. INSERT

overflow
Example of a dynamic table

1. INSERT
2. INSERT
3. INSERT
4. INSERT
5. INSERT
6. INSERT
7. INSERT

Worst-case analysis

Consider a sequence of \( n \) insertions. The worst-case time to execute one insertion is \( \Theta(n) \). Therefore, the worst-case time for \( n \) insertions is \( n \cdot \Theta(n) = \Theta(n^2) \).

\textbf{WRONG!} In fact, the worst-case cost for \( n \) insertions is only \( \Theta(n) \neq \Theta(n^2) \).

Let’s see why.

Tighter analysis

Let \( c_i \) = the cost of the \( i \)th insertion
\[
= \begin{cases} 
  i & \text{if } i - 1 \text{ is an exact power of 2}, \\
  1 & \text{otherwise}.
\end{cases}
\]

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
\text{size}_i & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 8 & 8 & 16 & 16 \\
\hline
\text{c}_i & 1 & 2 & 3 & 5 & 1 & 1 & 9 & 1 \\
\hline
\end{tabular}
Tighter analysis

Let $c_i =$ the cost of the $i$th insertion
\[
= \begin{cases} 
  i & \text{if } i - 1 \text{ is an exact power of } 2, \\
  1 & \text{otherwise.}
\end{cases}
\]

\[
\begin{array}{c|cccccccccc}
 i & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
 size_i & 1 & 2 & 4 & 4 & 8 & 8 & 8 & 8 & 16 & 16 \\
 c_i & 1 & 2 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

Tighter analysis (continued)

Cost of $n$ insertions = \[\sum_{i=1}^{n} c_i\]
\[\leq n + \sum_{j=0}^{\lfloor \log(n-1) \rfloor} 2^j\]
\[\leq 3n\]
\[= \Theta(n)\]

Thus, the average cost of each dynamic-table operation is $\Theta(n)/n = \Theta(1)$. 