A more extensive example of amortized analysis -

dynamic tables - data structure that expands
and contracts as items are inserted and
deleted.

Implemented with some kind of static data
structure (e.g., array, stack).

For now, just consider insertions.

\[
\text{table - insert } (T, x) : \\
\quad \text{ if } \text{size} (T) = 0 \text{ then } // \text{true initially} \\
\quad \quad \text{ allocate a table w/ 1 slot} \\
\quad \quad \text{size} (T) := 1 \\
\quad \quad \text{ if } \text{num} (T) = \text{size} (T) \text{ then } // \text{table is full} \\
\quad \quad \quad \text{ allocate a new table w/ 2 \cdot \text{size} (T) slots} \\
\quad \quad \quad \text{ insert all items in old table into} \\
\quad \quad \quad \quad \text{new table} \\
\quad \quad \quad \text{rename new table as } T \\
\quad \quad \quad \quad \text{size} (T) := 2 \cdot \text{size} (T) \\
\quad \quad \quad \quad \text{insert } x \text{ into table} \\
\quad \quad \quad \quad \text{num} (T) := \text{num} (T) + 1
\]

measure runtime in terms of number
of insertions performed on the
representation table

Naive idea: consider n table-insert ops.
Each takes at most O(n) insertions on the
underlying table \( \Rightarrow O(n^2) \),
but this is unduly pessimistic - expansion only happens rarely. Use amortized analysis to get better bound.

**Aggregate method:**

\[
\begin{align*}
\text{cost of } i^{\text{th}} \text{ operation:} \\
c_i &= \begin{cases} \\
 5 & \text{if } i-1 \text{ is a power of } 2 \\
 1 & \text{otherwise}
\end{cases}
\end{align*}
\]

So total cost of \( n \) ops is:

\[
\sum c_i = 1 + 2 + 3 + 1 + 5 + 1 + 1 + 1 + 9 + \ldots
\]

\[
= (1+1+\ldots+1) + (1+2+4+8+\ldots)
\]

\[
\leq n + 2n, \quad \text{since largest possible term in } 2^{nd} \text{ sum } \leq n
\]

\[
= 3n.
\]

\( \therefore \) Amortized cost of a single op. is 3.

**Accounting method:**

Gives some intuition for why amortized cost per op. should be 3:

1. for first insertion
1. for being copied into a bigger table
1. to pay for copying older ells in the table.

**Ex:**

\[
\begin{array}{c}
2 \\
\hline
a
\end{array} \rightarrow
\begin{array}{c}
1 \\
\hline
a
\end{array} \rightarrow
\begin{array}{c}
2 \\
\hline
b \quad \quad c \\
\hline
\end{array} \rightarrow
\begin{array}{c}
2 \\
\hline
a \quad b \quad c \\
\hline
\end{array} \rightarrow
\begin{array}{c}
2 \\
\hline
a \quad b \quad c \quad d \\
\hline
\end{array} \rightarrow
\begin{array}{c}
2 \\
\hline
a \quad b \quad c \quad d \quad e \quad f \\
\hline
\end{array} \rightarrow
\begin{array}{c}
2 \\
\hline
a \quad b \quad c \quad d \quad e \quad f \quad g \\
\hline
\end{array} \rightarrow
\begin{array}{c}
2 \\
\hline
\end{array}
\]

Use extrato copy a row next time.

Use extrato copy a row.
Potential method:

Idea: at immediately after an expansion, builds to half size when table is full to pay for next expansion.

Let $T_i = \text{table after } i^{th} \text{ insertion.}$

define $\Phi(T_i) = 2 \cdot \text{num}_i - \text{size}_i$

# Elts. in $T_i$  size of implementation table

Check:
- $\Phi(T_0) = 2 \cdot 0 - 0 = 0$
- $\Phi(T_i) \geq 0$ since as soon as an expansion occurs, the new rep. table is half full and grows from there.

Calculate amortized cost per op:

Case 1: $i^{th}$ op. does not trigger an expansion.

$$\hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1})$$

$$= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1})$$

$$= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1})$$

(num increases by 1, size is unchanged)

$$= 3.$$

Case 2: $i^{th}$ op. triggers an expansion

$$\hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1})$$

$$= \text{num}_i + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1})$$

$\uparrow$

copy i-1 elts.

$\uparrow$

twice

than insert

$\uparrow$

the old size

$\uparrow$

one less

than new

$\uparrow$

same

num

num
\[= \text{num}_i + (2 \cdot \text{num}_i - 2(\text{num}_{i-1}) - (2(\text{num}_{i-1}) - (\text{num}_{i-1}))\]
\[= 3\]

Now consider what happens with deletions also. If the table gets sufficiently empty, contract it to save space.

What should be the criterion for contracting? Want to avoid thrashing behavior. Rule is: halve the table size when the table is less than a quarter full.

Use potential method to analyze cost of a sequence of \( n \) inserts + deletes.

Define
\[\Phi(T_i) = \begin{cases} 2 \cdot \text{num}_i - \text{size}_i & \text{if } T_i \text{ is at least half full} \\ \frac{1}{2} \cdot \text{size}_i - \text{num}_i & \text{otherwise} \end{cases} \] (new clause)

Check:
- \( \Phi(T_0) = 0 \) (define table of size 0 to be full)
- \( \Phi(T_i) \geq 0 \) (check with simple algebra)

Calculate amortized cost per operation:

for insert:
- If table is \( \geq \frac{1}{2} \text{ full} \) before op, same analysis as before.
- Suppose table is \( < \frac{1}{2} \text{ full} \) before op.
Case 1: Table is still half full after op.

\[ \hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1}) \]

\[ = 1 + \left( \frac{1}{2} \cdot \text{size}_i - \text{num}_i \right) - \left( \frac{1}{2} \cdot \text{size}_{i-1} - \text{num}_{i-1} \right) \]

\[ = 1 + \left( \frac{1}{2} \cdot \text{size}_i - \text{num}_i \right) - \left( \frac{1}{2} \cdot \text{size}_{i-1} - \text{num}_{i-1} - 1 \right) \]

Since size doesn't change and num increases by 1

\[ = 0 \]

Case 2: Table is ≥ half full after op.

\[ \hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1}) \]

\[ = 1 + 2 \cdot \text{num}_i - \text{size}_i - \left( \frac{1}{2} \cdot \text{size}_{i-1} - \text{num}_{i-1} \right) \]

\[ = 1 + 2 \cdot (\text{num}_{i-1} + 1) - \text{size}_i - \left( \frac{1}{2} \cdot \text{size}_{i-1} - \text{num}_{i-1} \right) \]

Since size doesn't change and num increases by 1

\[ = 3 + 3 \cdot \left( \text{num}_{i-1} - \frac{1}{2} \cdot \text{size}_{i-1} \right) \]

\[ < 3 \quad \text{since table is < } \frac{1}{2} \text{ full before op} \]

For delete:
Suppose table is < \frac{1}{2} full before op.

Case 1: delete does not trigger a contraction.

\[ \hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1}) \]

\[ = 1 + \left( \frac{1}{2} \cdot \text{size}_i - \text{num}_i \right) - \left( \frac{1}{2} \cdot \text{size}_{i-1} - \text{num}_{i-1} \right) \]

\[ = 1 + \left( \frac{1}{2} \cdot \text{size}_i - \text{num}_i \right) - \left( \frac{1}{2} \cdot \text{size}_{i-1} - \text{num}_{i-1} \right) \]

\[ = 2 \]

Case 2: delete triggers a contraction.
\[ \hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1}) \\
= (1 + \text{num}_i) + (\frac{1}{2} \cdot \text{size}_{i-1} - \text{num}_{i-1}) - (\frac{1}{2} \cdot \text{size}_{i-1} - \text{num}_{i-1}) \\
\]

Since one elt. is deleted and num. are transferred

\[ = 1 + \frac{1}{4} \cdot \text{size}_{i-1} - \frac{1}{2} \cdot \text{size}_{i-1} + \text{num}_{i-1} \]

Since size is reduced by half

\[ = 1 + \text{num}_{i-1} - \frac{1}{4} \cdot \text{size}_{i-1} \]

\[ = 0 \quad \text{since table contracted, it was } \frac{1}{4} \text{ full.} \]

Suppose table is \( \geq \frac{1}{2} \) full before op.

Then no contraction occurs.

\[ \hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1}) \\
= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot \text{num}_{i-1} - \text{size}_{i-1}) \\
= 1 + (2 \cdot \text{num}_i - \text{size}_i) - (2 \cdot (\text{num}_i + 1) - \text{size}_i) \\
\]

Since size doesn't change, num decreases by 1

\[ = -1 \]

\[ \therefore \text{The amortized costs of all ops. are } O(1). \]

And total cost of a sequence of \( n \) ops is \( O(n) \).