Today

- Amortized Analysis
- Example: Dynamic Hash Tables
- Methods: Aggregate Accounting, Potential Method

Dynamic Hashing
- Recall task of building hash tables.
  Step 0: Initialize $T \leftarrow$ Hash size ($m$)

- But what should $m$ be?
  - Expected size of table.
  - What if this is not known?
Dynamic Table:

- Guess size of table: initially 1;
- Each time # elements exceeds current guess:
  - Double guess;
  - Destroy old table;
  - Build new one;

Analysis:
- Worst case insertion time = \( \Theta(n) \) for
  \( n^{th} \) insertion.
- Total cost = \( \Theta(n^2) \) ...
  \[ \text{WRONG!} \]

While some insertions cost \( \Theta(n) \),
most cost \( \Theta(1) \);
Correct Analysis

Cost of $i^{th}$ insertion = $\Theta(1)$ if $i \neq 2^k$

= $\Theta(i)$ if $i = 2^k + 1$

Total cost of $n^{th}$ insertions

\[
\sum_{k=0}^{\lfloor \log n \rfloor} \sum_{i=1}^{2^k} (2^k + 1) + \sum_{i=2^k+1}^{n} i
\]

\[
= 2^{\lfloor \log n \rfloor + 1} + n
\]

= $\Theta(n)$

Amortized Analysis:

The amortized cost of an operation is $C$ if every sequence of $n$ operations takes at most $n \cdot C$ time.
Aside On Terminology

Amortized means averaged in English... but for us it carries special meaning.

Average Time: running time, averaged over a distribution of inputs.

Expected Time: running time, averaged over algorithm's random coins.

Amortized Time: running time, averaged over a sequence of operations.

(No assumption on inputs! No randomizers needed in algorithm.)

In example of dynamic hash tables, we say that the amortized running time for an insertion is $\Theta(1)$. 
Techniques for Amortized Analysis

1. Aggregate Method
2. Accounting Method
3. Potential Method

Aggregate Method:

1. Just add up cost of \( n \) operations
2. Divide by \( n \) to get amortized cost.

- Just saw an example...
- Often hard to do step 1. without further guidance, so other methods more widely applicable.

Accounting Method:

0. Start with a bank balance of 0.
1. For each operation charge a fixed cost \( c \) my be negative
2. If \( i^{th} \) operation takes time \( t_i \), then add \( (C - t_i) \) to the bank.
3. Show balance never negative.
Claim: Accounting Method implies

\[ \text{amortized cost} \leq C \]

Proof:

\[ \sum_{i=1}^{n} (C-t_i) \geq 0 \]

\[ \Rightarrow n \cdot C - \sum_{i=1}^{n} t_i \geq 0 \]

\[ \Rightarrow n \cdot C \geq \sum_{i=1}^{n} t_i \]

\[ \Rightarrow C \geq \frac{\sum_{i=1}^{n} t_i}{n} \]

\[ \Rightarrow C \geq \frac{\sum_{i=1}^{n} t_i}{n} \]

Accounting analysis for dynamic tables:

- Charge \( C = 3 \) $ for each insert.
- For typical insert 1 $ pays for current work, while 2 $ added to bank account.
- For \( 2^k \) inserts bank balance pays for
- destroying & creating new table, and becomes zero after more.
Analysis

Typical picture

Bank balance

Insert: $3 = 1 + 2$

Pxyz for current work.

Special Insert

Invariant: Bank balance $\geq 0$.

1. Associate potential function $\Phi$ with data structure.

2. Show $\Phi$ always non-negative.

3. Initial potential $= 0$.

4. For every step show

$$\Phi_i \leq \Phi_{i-1} - t_i + C$$

Claim: implies amortized cost $\leq C$.

Proof: Have $t_i \leq C + \Phi_{i-1} - \Phi_i$

$$\Rightarrow \sum_{i=1}^{n} t_i \leq n \cdot C + \Phi_0 - \Phi_n$$

$$= n \cdot C - \Phi_n \quad [\Phi_0 = 0]$$

$$\leq n \cdot C \quad [\Phi_n > 0]$$

$\blacksquare$
Difference between potential and bank balance...

- potential only depends on current state of the data structure, not how we reached this state
- bank balance may depend on sequence of operations...

Thus Potential Method is a special case of the Accounting Method.

---

Potential Analysis of Dynamic Hash Table.

1. $\Phi(i) = 2i - 2^{\lceil \log i \rceil}$

2. $2^{\lceil \log i \rceil} \leq 2^{(\log i) + 1} = 2 \cdot 2^\log i = 2i$

$\Rightarrow \Phi(i) > 0$
3. \( \phi(0) = 0 - 2^{\log^0} = 0 \)

\[ \text{Assume } 2^{\log x} = x \pm x. \]

4. **Crucial Step**

**Insertion:**

\( i = 2^k \): \( t_i = 1 \)

\[ c = 3 \]

\[ \phi_i = 2i - 2 \]

\[ = \phi_{i-1} + 2 \quad \checkmark \]

\( i = 2^{k+1} \): \( t_i = i \)

\[ c = 3 \]

\[ \phi_i = \times 3 \]

\[ \phi_{i-1} = 1 + 2(i-1) - (i-1) = i \quad \checkmark \]

have \( \phi_i \leq \phi_{i-1} - t_i + c \quad \checkmark \)
Moral Of This Story

- Amortized cost often less than max. cost; good enough to estimate actual cost of repeated use.

- Potential / Other accounting method give way to estimate this cost.