Amortized Analysis

Definition
The **amortized cost** of a sequence of operations is the total cost divided by the number of operations.

- Same as average-case?
- Compare to worst-case and best-case.
- Mainly a data structures issue?

Binary Counter

Problem
Increment a bit-string from an integer $k$ to $k + 1$

- Best-case cost:
- Worst-case cost:

**Amortized version**: Consider $n$ consecutive increments.

Amortized Binary Counter

<table>
<thead>
<tr>
<th>Counter</th>
<th>Update cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>How often does 1 bit change?</td>
</tr>
<tr>
<td>0001</td>
<td>How often do 2 bits change?</td>
</tr>
<tr>
<td>0010</td>
<td>How often do 3 bits change?</td>
</tr>
<tr>
<td>0011</td>
<td>...</td>
</tr>
<tr>
<td>0100</td>
<td>Total cost for $n$ increments:</td>
</tr>
<tr>
<td>0101</td>
<td>Amortized cost for each increment:</td>
</tr>
<tr>
<td>0110</td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
</tr>
</tbody>
</table>
Self-Organizing Search

Problem
Keep reordering an array to optimize a sequence of linear searches.

- Supports what ADT?
- Parameters: $n$ distinct elements, $m > n$ searches
- Request frequencies: $k_1 \geq k_2 \geq \cdots \geq k_n$
- How to analyze an approach?

Approaches

- **Optimal Static Ordering (OPT)**
  Look at the entire request sequence and determine frequencies. Order elements by decreasing frequency.

- **Move-to-front (MTF)**
  Move each element to the front after it is accessed.

- **Transpose**
  Swap each element with its predecessor after it is accessed.

Which way is best?

Competitive Analysis

**Idea:** Don’t analyze worst-case, compare to optimal!

- Transpose:

- MTF: