Amortized Analysis

Definition
Methods
Incrementing a Bit String
Dynamic Tables

Amortized Analysis
Methods of Amortized Analysis
Incrementing a Bit String
Bit Increment, Aggregate Method
Bit Increment, Accounting Method
Bit Increment, Potential Method
Dynamic Tables
Dynamic Tables, Aggregate Method
Dynamic Tables, Accounting Method
Dynamic Tables, Potential Method
Dynamic Tables Deletion

Amortized Analysis
Methods of Amortized Analysis

Amortized Analysis
In an amortized analysis, we consider the total time of a sequence of operations. Even if a single operation is $O(f(n))$, the average of $n$ operations might be $o(f(n))$.

Examples:
- Sequence of heapify calls from build-heap.
- Sequence of calls to tree-successor.
- Repeatedly incrementing a bit string.
- Insertions and deletions on a dynamic table.
- make-set, union, and find-set.

Methods of Amortized Analysis

Aggregate Method: Directly analyze total time.

Accounting Method:
Assign an amortized cost to each operation.
Show total amortized cost $\geq$ total time.

Potential Method:
Specify a potential function $\Phi_i$ after $i$ operations.
Show amortized cost of operation $i = \frac{\text{time of operation } i + \Phi_i - \Phi_{i-1}}{\Phi_i}$
Total amortized cost $= \text{total time} + \Phi_n - \Phi_0$
Show $\Phi_i - \Phi_0 \geq 0$ for all values of $i$.
Implies total amortized cost $\geq$ total time.
**Incrementing a Bit String**

Suppose $A$ is a bit string with $A[0]$ as the lowest order bit. Analyze total number of bit flips after $n$ increments.

**INCREMENT(A)**

$$
i \leftarrow 0$$

while $i < A.length$ and $A[i] = 1$

$$A[i] \leftarrow 0$$

$$i \leftarrow i + 1$$

if $i < A.length$

then $A[i] \leftarrow 1$

---

**Bit Increment, Aggregate Method**

Aggregate Method:

Assume $n$ increments starting from all 0s.

$A[0]$ flips every increment for $n$ flips.

$A[1]$ flips every 2nd time for $\leq n/2$ flips.


$A[i]$ flips every $2^i$th time for $\leq n/2^i$ flips.

Number of flips $\leq n + \frac{n}{2} + \frac{n}{4} + \ldots$

$$= n \left(1 + \frac{1}{2} + \frac{1}{4} + \ldots\right)$$

$$= 2n \text{ which is } O(n)$$

---

**Bit Increment, Accounting Method**

Accounting Method:

Assume $n$ increments starting from all 0s.

INCREMENT flips exactly one bit from 0 to 1.

Assign an amortized cost of 2 units/increment.

Both units are assigned to the bit that is flipped from 0 to 1.

Use one unit immediately for flip from 0 to 1.

Save other unit for when it is flipped back to 0.

All bit flips are accounted for, so the total amortized cost of $2n$ is $\geq$ number of bit flips.

---

**Bit Increment, Potential Method**

Potential Method:

Assume $n$ increments starting with all 0 bits.

Let $A_i =$ bit string after the $i$th increment.

Let $\Phi_i =$ number of bits in $A_i$ equal to 1.

Use an amortized cost of 2 units/increment because $2 = \text{bit flips} + \Phi_i - \Phi_{i-1}$

All bit flips are accounted for, so total amortized cost of $2n = \text{total + change in potential}$.

Total time is $\leq 2n$ because $\Phi_i - \Phi_0 \geq 0$ for all $i$. 
Dynamic Tables

Assume \( T.num \) is number of elements in \( T \).
Assume \( T.size \) is number of slots in \( T \).
Assume \( T.num = 0 \) and \( T.size = 1 \) initially.

**Table-Insert** \((T, x)\)
- if \( T.num = T.size \)
  - then reallocate \( T \) with size \( 2 \cdot T.size \)
  - \( T.size \leftarrow 2 \cdot T.size \)
  - \( T.num \leftarrow T.num + 1 \)
  - insert \( x \) into \( T \)

Analyze number of insertions + amount of copying during reallocations.

Dynamic Tables, Aggregate Method

Aggregate Method:

Assume \( n \) insertions starting from 0 elts.
Assume \( T.num \) time for reallocating \( T \).
Assume 1 unit time for rest of **Table-Insert**.

Reallocate when \( T.num \) is a power of 2.
Let \( S \) = the powers of 2 that are \( \leq n \).
\( n \geq \) largest value in \( S \). \( n/2 \geq \) 2nd largest value.
\( n/4 \geq \) 3rd largest value, etc.

\[
\sum_{s \in S} s \leq n + n/2 + n/4 + \ldots \leq 2n
\]

So insertions + reallocation time \( \leq n + 2n = 3n \)

Dynamic Tables, Accounting Method

Accounting Method:

Let \( x_i \) be the \( i \)th element that is inserted.
Assign an amortized cost of 3 units/insertion.
Use one unit immediately for inserting \( x_i \).
Save two units for future reallocation:
- one for \( x_i \) and the other for \( x_{i-s/2} \)
  where \( s = T.size \).
When reallocating, all elts are accounted for, so the total cost of \( 3n \) is \( \geq \) time units.

Dynamic Tables, Potential Method

Potential Method:

Let \( T_i \) be the table after \( i \) elements are inserted.

Let \( \Phi_i = 2 \cdot T_i.num - T_i.size = 2 \cdot i - T_i.size \)
Assign an amortized cost of 3 units/insertion.

If no reallocation, cost is 1 and
\( 3 = 1 + \Phi_i - \Phi_{i-1} \) because \( \Phi_i = 2 + \Phi_{i-1} \).

If a reallocation, cost is \( i \) (need to copy \( i - 1 \) elts):
\( 3 = i + \Phi_i - \Phi_{i-1} \) because
\( \Phi_i = 2 \cdot i - 2 \cdot (i - 1) = 2 \) and
\( \Phi_{i-1} = 2 \cdot (i - 1) - (i - 1) = i - 1 \).
\( \Phi_i - \Phi_0 \) is always \( \geq 0 \) so \( 3n \geq \) total cost.
Dynamic Tables Deletion

Suppose deletions are allowed.

\[ T\text{.size} \leftarrow T\text{.size}/2 \] when array is 1/4 full.

Assign an amortized cost of 2 units/deletion.

Use one unit immediately for deleting \( x_t \).
Save one unit for reallocation of \( x_{t-s/4} \),
where \( s = T\text{.size} \).

If space is at a premium,
then use a expansion/contraction factor < 2.
In the analysis, each operation will cost more.