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.
Repeated insertions into a dynamic array.
MAKE-SET, UNION, and FIND-SET.

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 an initial charge (potential).
Amortized op cost = time + change in potential.
Total amortized cost = total time + change in potential.
Show change ≥ 0 or potential ≥ 0 at all times.

Incrementing a Bit String

A is a bit string. Analyze number of bit flips.

$\textsc{Increment}(A[0 \ldots m-1])$
  $i \leftarrow 0$
  while $i < m$ and $A[i] = 1$
    do $A[i] \leftarrow 0$
      $i \leftarrow i + 1$
  if $i < m$
    then $A[i] \leftarrow 1$

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.
$A[2]$ flips every 4th time for $\leq n/4$ flips.
$A[i]$ flips every 2^ith time for $\leq n/2^i$ flips.

$$\begin{aligned} \text{Number of flips} &\leq n + \frac{n}{2} + \frac{n}{4} + \ldots \\ &= n \sum_{i=0}^{\infty} \frac{1}{2^i} \\ &= 2n \in O(n) \end{aligned}$$

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 cost of $2n$ is $\geq$ number of bit flips.

Potential Method:

Assume n increments starting with k bits $= 1$.

Let potential $=$ number of bits equal to 1.

Use an amortized cost of 2 units/increment because $2 =$ bit flips $+$ (bit flips from 0 to 1 $-$ bit flips from 1 to 0)

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

Total time is $\leq 2n + k$ because potential ≥ 0.

Dynamic Arrays

Assume $num[A]$ is number of elements in A.
Assume $num[A] = 0$ and $size[A] = 1$ initially.

ARRAY-INSERT(A, x)
  if $num[A] = size[A]$
    then reallocate A with length $2 \cdot size[A]$
      $size[A] \leftarrow 2 \cdot size[A]$
  $num[A] \leftarrow num[A] + 1$
  $A[num[A]] \leftarrow x$

Analyze number of insertions + amount of copying during reallocations.

Aggregate Method:

Assume n insertions starting from 0 elts.

Assume $num[A]$ time for reallocating A.

Assume 1 unit time for rest of ARRAY-INSERT.

Reallocate when $num[A]$ is a power of 2.

$$\sum_{i=0}^{\lfloor \lg n \rfloor} 2^i \leq n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n$$

So insertion + reallocation time $\leq n + 2n = 3n$

Accounting Method:

Assign an amortized cost of 3 units/insertion.

Use one unit immediately for inserting $A[j]$.

Save two units for future reallocation:
one for $A[j]$ and the other for $A[j - s/2]$,
where $s = size[A]$.

When reallocating, all elts are accounted for,
so the total cost of $3n$ is $\geq$ time units.

Suppose deletions are allowed.

$size[A] \leftarrow size[A]/2$ when array is 1/4 full.

Assign an amortized cost of 2 units/deletion.

Use one unit immediately for deleting $A[j]$.
Save one unit for reallocation of $A[j - s/4]$,
where $s = size[A]$.

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