Amortized Analysis

- suppose we want to perform a sequence of n operation on some dynamic data structure
  - worst-case cost of a single operation is $O(f)$
- a naive analysis give a total worst-case of $O(nf)$ for all n operations
- If the worst-case cannot occur on every operation due to some invariant this may be a huge overestimation

- Amortized analysis is a way of proving that ewen is some operation is occasionally expensive its cost is offset by other cheaper operations

Amortized Bound

$$\sum \text{amortized cost} \geq \sum \text{actual cost}$$

Amortized cost must preserve the ~~actual cost~~ total cost.

Binary Counter

A is an arbitrarily long array of bits

```
INCREMENT(A)
i = 0
while A[i] == 1
    A[i] = 0
    i = i + 1
A[i] = 1
```

0 1 1 0 1 1 1

0 1 1 1 0 0 0

We will track time as $O(\#$ of bit flips$)$

- This depends on the position of the rightmost 0 in A

- If the first k bits are all ones then Increment takes $O(k)$
- The ~~bary~~ binary representation of some integer n takes $\lfloor \lg n \rfloor + 1$
- If A represents a number between 0 and n Increment takes $O(\log n)$ in the worst case
- Suppose we invoke Increment n times in a row starting with all 0's
- Naive bound $O(n \log n)$
- Actual time $\Theta(n)$

Aggregate Method

- assuming we know the sequence of n operations.
- determine the worst case cost for each one
- add these up and divide by n

Binary Counter

- Notice that each bit does not flip every time.
    - Bit A[0] flips every time
    - Bit A[1] flips every other time
    - Bit A[2] flips every 4^{th} time
- In general bit A[i] every ~~2~~ i^{th} 2^i th time

- Starting with an array of all 0's, a sequence of n calls to increment will flip $A[i]$ exactly $\lfloor \frac{n}{2^i} \rfloor$ times

- Total number of bit flips

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor < \sum_{i=0}^{\infty} \frac{n}{2^i} = 2n$$

- So, on average, each call to increment flips fewer than 2 bits

- The overall sequence takes $O(n)$-time

- The amortized cost of each call to increment is $O(1)$-time

Accounting Method (taxation or Bankers)

- the idea is that we assign different ~~costs~~ amortized amounts to different operations
- the amortized cost a_i may be more or less than the actual cost c_i
- If $a_i > c_i$ the difference is stored as credit
- If $a_i < c_i$ ~~t~~here must be enough credit to cover the difference
- for any sequence we must have

$$\sum_{i=1}^{n} a_i \geq \sum_{i=1}^{n} c_i$$

Binary Counter

Key observations

- starting from all 0's a bit can only flip from a 1 to a 0 if it was previously flipped from a 0 to a 1
- each call ~~to~~ Increment flips exactly one bit from 0 to 1

- We set amortized costs as follows
  - flipping a bit from 0 to 1 costs \$2
  - flipping a bit from 1 to 0 costs \$0

- When we flip from 0 to 1 we pay \$1 for the flip and store \$1 credit on that bit to later pay for flipping it back.

- each 1 bit in A stores \$1 credit
- the number of 1 bits is never negative
- the total amortized cost for n calls to Increment is just $2n$ is a valid bound on the total actual cost which is $O(n)$

Potential Method (physicists)

- instead of associating credit with individual pieces of data we assign the prepaid work to the data structure as a whole.
- let Φ_i be the potential of the data structure after the i^{th} operation.
- The amortized cost of the i^{th} operation is

$$a_i = c_i + \Phi_i - \Phi_{i-1}$$

- So the amortized cost of n operations is the actual cost plus the total increase in potential

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} (c_i + \Phi_i - \Phi_{i-1}) = \sum_{i=1}^{n} c_i + \Phi_n - \Phi_0$$

- A potential function is <u>valid</u> if $\Phi_i \geq \Phi_0$ for all i

- If the potential function is valid the the total amortized cost bounds the total actual cost

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} a_i - (\Phi_n - \Phi_0) \leq \sum_{i=1}^{n} a_i$$

Binary Counter

- generalize the accounting method to create our potential function

- let Φ_i be the number of 1 bits in A after the i^{th} call to Increment

- Initially all bits are 0 so $\Phi_0 = 0$

- ~~$\Phi_i \geq 0$~~

- $\Phi_i > 0$ for all $i > 0$ so this is a valid potential function.

- The actual cost of the i^{th} call to Increment

c_i = # of bits flipped from 0 to 1 +
# of bits flipped from 1 to 0

- So the amortized cost is

let x = # of bits flipped from 0 to 1
let y = # of bits flipped from 1 to 0

$$a_i = c_i + \Phi_i - \Phi_{i-1}$$

~~= 2 × # of bits flipped from 0 to 1~~

$$= x + y + x - y$$

$$= 2x$$

$$= 2$$