Amortized Analysis

Data structure: $\theta_1, \theta_2, \theta_3, \ldots, \theta_n$

Worst case cost of sequence

Cost for $\theta_i$ is $O(c)$

=> Cost of sequence is $O(nc)$ - cannot be $O(n)$

\[
\sum_{i=1}^{n} \text{cost}(\theta_i) = \text{Average cost of an operation}
\]

Worst case

Stack: push, pop

\[
O(1) \Rightarrow \text{cost}(\text{req}) = \Theta(n)
\]

$\text{makePop}(h) - \text{remove top } h \text{ elements} - O(h)$

$\text{O(n)}$

Amortized costs

$\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n$

\[
\text{cost}(\text{req}) = n \times O(n) = O(n^2)
\]

Account's method

$\$2 \text{/push}$

$\$0 \text{/pop}$

\[
\$2n
\]

Potential function method

Bank = $51$ (fee per stack

Account = $-2$

$\Phi(A)$

$\Phi(D)$

$\Phi(\text{data structure})$

$\Phi(\text{stack}) = \# \text{of items on stack}$
\( \Phi(\text{stack}) - \# \text{items on the stack} \)

\[
\sum_{i=1}^{n} \text{actual cost}(\theta_i) = \text{amortized cost}(\theta_i) + \text{amortized cost}(\theta_i) + \text{amortized cost}(\theta_i) + \cdots + \text{amortized cost}(\theta_i)
\]

\[
\sum_{i=1}^{n} \text{amortized cost}(\theta_i) + \sum_{i=1}^{n} \Phi(\text{before } \theta_i) - \Phi(\text{after } \theta_i)
\]

\[
\Phi(\text{before } \theta_i) - \Phi(\text{after } \theta_i) - \Phi(\text{initial}) - \Phi(\text{final})
\]

\[
\Phi(\text{after } \theta_n) - \Phi(\text{after } \theta_{n-1}) - \Phi(\text{after } \theta_{n-2}) - \cdots - \Phi(\text{after } \theta_1)
\]

\[
\text{Total actual costs} = \text{Total amortized costs} + \boxed{\Phi(\text{initial}) - \Phi(\text{final})} \geq 0
\]

\[
\boxed{\text{Total amortized costs}} \leq \boxed{2n}
\]
For Figure 17.2 An 8-bit binary counter as its value goes from 0 to 16 by a sequence of 16 INCREMENT operations. Bits that flip to achieve the next value are shaded. The running cost for flipping bits is shown at the right. Notice that the total cost is always less than twice the total number of INCREMENT operations.

Operations on an initially zero counter cause \( A[1] \) to flip \( \lfloor n/2 \rfloor \) times. Similarly, bit \( A[2] \) flips only every fourth time, or \( \lfloor n/4 \rfloor \) times in a sequence of \( n \) INCREMENT operations. In general, for \( i = 0, 1, \ldots, k - 1 \), bit \( A[i] \) flips \( \lfloor n/2^i \rfloor \) times in a sequence of \( n \) INCREMENT operations on an initially zero counter. For \( i \geq k \), bit \( A[i] \) does not exist, and so it cannot flip. The total number of flips in the sequence is thus

\[
\sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n,
\]

by equation (A.6). The worst-case time for a sequence of \( n \) INCREMENT operations on an initially zero counter is therefore \( O(n) \). The average cost of each operation, and therefore the amortized cost per operation, is \( O(n)/n = O(1) \).
Hash Table

```
0 1 2 ... a - 1 p - 1 p \prime
```

**Key-to-Address Function (Hash Function)**

\[ h(x) \equiv x \mod p \]

**Collision Resolution**