Amortized Analysis

- Time required to perform a sequence of data structure operations is averaged over all the operations performed.
- Aggregate Method.
- Accounting Method.
- Potential Method.

Aggregate Method

- Total cost of all operations is determined. Amortized cost is then the average.

Accounting Method

- Operations of a given type are overcharged early in the sequence.
- Amortized cost of these operations is therefore higher than the actual cost.
- Prepaid credit is allocated to specific objects in the data structure.
- The credit is later used on subsequent operations associated with these objects.
- Amortized cost of these operations is therefore lower than the actual cost. if the total credit associated with the objects is nonnegative at all times, the total amortized cost is an upper bound on the actual cost.

Potential Method

- The prepaid credit is associated with the data structure and can be used in connection with any subsequent operation.
- The credit can be look at as a potential of the data structure that can be released at later stage to pay for any operation.
- Amortized cost of an operation: actual cost plus increase of potential due to the operation.
- Total amortized cost for n operations:

$$\sum_{i=1}^{n} \bar{c_i} = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)$$

- If $\Phi(D_n) - \Phi(D_0) \geq 0$, then the total amortized cost is an upper bound on the total actual cost.

Aggregate Method - Modified Stacks

Stack operations:

- `push(x,S)`
- `pop(S)`
- `multipop(k,S)`

Worst-case time complexity of individual operations:

- `push(x,S)`: $O(1)$
- `pop(S)`: $O(1)$
- `multipop(k,S)`: $O(n)$

Amortized time complexity over a sequence of operations:

- An object can be pushed on the stack.
- An object on the stack can be popped only once (either by `pop(S)` or by `multipop(k,S)`).
- Any feasible sequence of n operations requires $O(n)$ time in total.
- The amortized cost of each operation is $O(1)$.

Aggregate Method - Binary Counters

```
00000000
00000001
00000010
00000011
00000100
00000101
00000110
00000111
00001000
```

- An increment can in worst case take $O(k)$ flips leading to a conclusion that n increments will require $O(nk)$ flips.
- Basic observation: k-th bit flips every 2^k-th time
- k-th bit flips $\lfloor n/2^k \rfloor$ times when n increments are performed. Bits higher than $\lfloor \log n \rfloor$ never flip.

$$\sum_{k=0}^{\lfloor \log n \rfloor} \lfloor n/2^k \rfloor < n \sum_{i=0}^{\infty} 1/2^i = 2n$$

Aggregate Method - Table Expansion

- Elements are inserted into a table in $O(1)$ time provided that there is available space. If all space used up, a new table, twice as big, is created, all ellements in the original table are moved and the new element is inserted.
- Worst-case analysis: each operation can take $O(n)$ time. Total worst-case time complexity: $O(n^2)$.
- The cost of the i-th operation:

$$c_i = \begin{cases} i & \text{if } i-1 \text{ is a power of } 2 \\ 1 & \text{otherwise} \end{cases}$$

$$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lfloor \log n \rfloor} 2^j < n + 2n = 3n$$

Accounting Method - Modified Stacks

- Initially the stack is empty, total credits are 0.
- When pushing, amortized cost is 2, one unit for pushing, one unit saved for later pop or multipop of the item.
- For any sequence of n opertions, the total amortized cost is nonnegative and it is an upper bound on the total actual cost.

Accounting Method - Binary Counters

- Actual cost of setting a bit to 0 or to 1 is 1.
- Amortized cost of setting a bit to 1 is 2.
- Amortized cost of setting a bit to 0 is 0.
- Amortized cost of incrementing is at most 2: only one bit is set to 1. Resetting to zero has been paid for in advance.
- Total credit is always non-negative (number of 1's is non-negative).

Accounting Method - Table Expansion

- Each inserted item pays 3 units: 1 for its own insertion, 1 to be moved when the table is expanded, and 1 for some other item to be moved when the table is expanded (for items that were already moved once).
- Total credit is always non-negative and it is an upper bound on the total actual cost.

Potential Method - Modified Stacks

- Potential function Φ: stack size.
- $\Phi(D_0) = 0$.
- $\Phi(D_i) - \Phi(D_0) = \Phi(D_i) \geq 0$
- Amortized cost of push is

$$\bar{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 1 = 2$$

- Amortized cost of pop is

$$\bar{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 - 1 = 0$$

- Amortized cost of multipop is

$$\bar{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = k - k = 0$$

- Total amortized cost of n operations is $O(n)$.
- The worst-case cost of n operation is $O(n)$.

Potential Method - Binary Counters

- Potential function Φ: number of 1's.
- Amortized cost of i-th operation:
  - Assume that t_i is reset
  - Actual cost is $t_i + 1$
  - Potential drops by $t_i - 1$
  - Amortized cost is $\bar{c_i} = t_i + 1 - t_i + 1 = 2$
- Total amortized cost of n increments is $O(n)$.
- Since $\Phi(D_0) = 0$, and potential is always non-negative, total amortized cost is an upper bound on the total actual cost.

Potential Method - Table Expansion

- $n(D_i)$ is the number of elements in D_i.
- $s(D_i)$ is the size of D_i.
- Potential is 0 right after the expension and build up until the table is full to cover for next expansion.

$$\Phi(D_i) = 2n(D_i) - s(D_i)$$

- amortized cost of simple insertion

$$\bar{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + (2n(D_i) - s(D_i)) - (2n(D_{i-1}) - s(D_{i-1})) = 3$$

- amortized cost of expanding insertion

$$\bar{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = n(D_i) + (2n(D_i) - s(D_i)) - (2n(D_{i-1}) - s(D_{i-1})) = 3$$

Potential Method - Table Expansion and Contraction

- Elements are inserted into a table in $O(1)$ provided that there is available space. If all space is used up, a new table twice as big is created, all ellements in the original table are moved and the new element is inserted.
- Elements are deleted in $O(1)$ time. If the used space falls below $1/4$ of the total space, a new table twice as small is created, all elements in the original table are moved.
- If $n(D_i) = 0$ then $s(D_i) = 0$.
- Potential is defined as follows.

Potential Method - Table Expansion and Contraction

$$\Phi(D_i) = \begin{cases} 2n(D_i) - s(D_i) & \text{if } \alpha(D_i) \geq \frac{1}{2} \\ \frac{s(D_i)}{2} - n(D_i) & \text{otherwise} \end{cases}$$

- Potential is 0 after the contraction and expansion

Potential Method - Table Expansion and Contraction - Insertions

- amortized cost of simple insertion:
- amortized cost of simple insertion when $\frac{n(D_{i-1})}{s(D_{i-1})} \geq \frac{1}{2}$:

$$\bar{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2n(D_i) - s(D_i) - 2n(D_{i-1}) + s(D_{i-1}) = 3$$

- amortized cost of simple insertion when $\frac{n(D_{i-1})}{s(D_{i-1})} < \frac{1}{2}$:
- if $\frac{n(D_i)}{s(D_i)} < \frac{1}{2}$:

$$\bar{c_i} = 1 + \frac{s(D_i)}{2} - n(D_i) - \frac{s(D_{i-1})}{2} + n(D_{i-1}) = 0$$

- if $\frac{n(D_i)}{s(D_i)} \geq \frac{1}{2}$:

$$\bar{c_i} = 1 + 2n(D_i) - s(D_i) - \frac{s(D_{i-1})}{2} + n(D_{i-1}) =$$

$$1 + 2n(D_{i-1}) + 1 - s(D_{i-1}) - \frac{s(D_{i-1})}{2} + n(D_{i-1}) =$$

$$3 + 3n(D_{i-1}) - 3\frac{s(D_{i-1})}{2} < 3 + 3s(D_{i-1}) - 3s(D_{i-1}) = 3$$

- amortized cost of expanding insertion (as without contractons):

$$\bar{c_i} = n(D_i) + (2n(D_i) - s(D_i)) - (2n(D_{i-1}) - s(D_{i-1})) = 3$$

Potential Method - Table Expansion and Contraction - Detions

- amortized cost of deletion
- amortized cost of deletion when $\frac{n(D_{i-1})}{s(D_{i-1})} < \frac{1}{2}$:
- no contraction:

$$\bar{c_i} = 1 + \frac{s(D_i)}{2} - n(D_i) - \frac{s(D_{i-1})}{2} + n(D_{i-1}) = 2$$

- contraction:

$$\bar{c_i} = n(D_i) + 1 + \frac{s(D_i)}{2} - n(D_i) - \frac{s(D_{i-1})}{2} + n(D_{i-1}) = 1$$

- since $\frac{s(D_i)}{2} = \frac{s(D_{i-1})}{4} = n(D_i) + 1$
- amortized cost of deletion when $\frac{n(D_{i-1})}{s(D_{i-1})} \geq \frac{1}{2}$ is also bounded from above by a constant (exercise).

Potential Method - Convex Hulls

- Sort from left to right, $\Theta(n \log n)$.
- Insert from left to right.
- Insertion of i-th point can require $O(i)$ time.
- $\Phi(D_i)$: number of points on the convex hull
- Amortized insertion time is the actual insertion time (number of points removed + 3) plus potential change (1-number of points removed).
- Amortized time complexity is $O(n)$ leading to another $\Theta(n \log n)$ algorithm.