Analysis of Algorithms

Amortized Analysis

Outline

- Definition
- Aggregate Method
- Accounting Method
- Potential Method
- Examples : Binary Counter and Dynamic Table

Cost of A Sequence of Operations

- Let operation A requires $\Theta(n)$ cost in worst-case
- Calling A m times costs $\Theta(mn)$?
- Not necessary : it may cost $O(mn)$
- Sometimes worst cases do not happen consecutively in a sequence of calls
- Actual worst-case cost may be $o(mn)$

Amortized Analysis

- The worst-case cost for any sequence of m operations
- Average performance of each operation in worst case (no probability is involved).

$$\text{amortized time} = \frac{\text{worst time for a sequence of } m \text{ ops}}{m}$$

Example

- Given a list of n elements
- Sort this list m times using InsertionSort
- First time : worst-case $\Theta(n^2)$
- 2^{nd} - m^{th} times : worst-case $\Theta(n)$
- Total worst-case time : $\Theta(n^2 + mn)$
- Amortized time : $\Theta(n^2 / m + n)$

Example

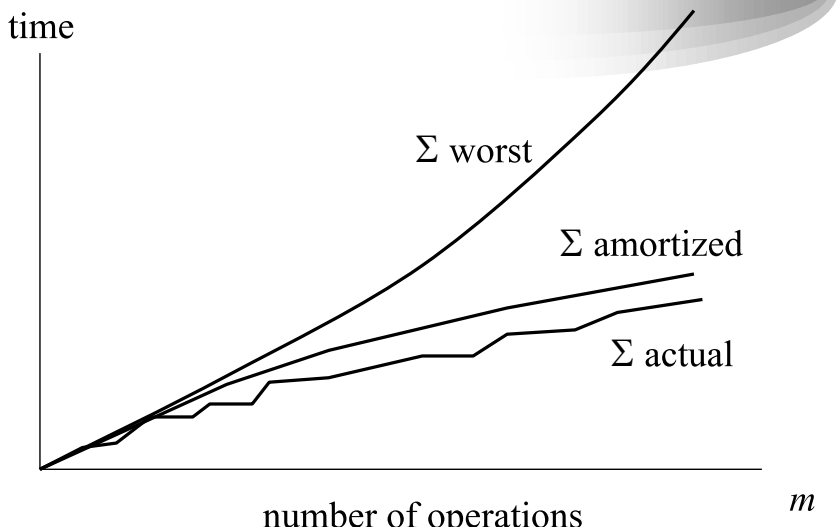
- Given a list of n elements
- Sort this list m times using SelectionSort
- First time : worst-case $\Theta(n^2)$
- 2^{nd} - m^{th} times : worst-case $\Theta(n^2)$
- Total worst-case time : $\Theta(mn^2)$
- Amortized time : $\Theta(n^2)$

Disjoint Sets

- Union : Union by rank : O(1)
- Find : Path compression : $O(\log n)$
- A sequence of m Union and Find operations perform on n elements : $O(m \log^* n)$
- Amortized cost : $O(\log^* n)$

$$\log^* 2^{2^{2^2}} = 4$$

Amortized Analysis



Amortized Analysis Techniques

- Aggregate Method
- Accounting Method
- Potential Method

Aggregate Method

- Compute the worst-case time $T(m)$ in total of a sequence of m operations
- Amortized cost $= T(m) / m$
- Amortized cost of each operation is the same even when there are several types of operations in the sequence.

Accounting Method

- Assign (guess) amortized cost for each operation
- If an operation's actual cost is less than its amortized cost, the difference is assigned to specfic objects in the data structure as *credit*
- Credit can be used later for operations whose actual cost exceeds their amortized cost
- Correct if the credit is nonnegative at all times

Potential Method

- Assign (guess) a potential function for the entire data stucture
- Potential energy increases if amortized cost exceeds actual cost
- Potential energy decreases if amortized cost is less than actual cost
- Amortized cost = Actual cost + Δ(potential)
- Correct if potential is never less than its initial

Binary Counter

- a *k*-bit binary counter A[0..k-1]
- A[k-1] : MSB, A[0] : LSB
- count upward from 0 *m* times



0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0

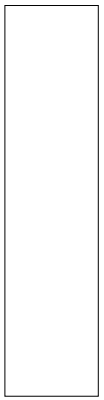
Increment Binary Counter

```
Increment( A[0..k-1] )
{
  i = 0
  while ( i < k and A[i] = 1 )
    A[i] = 0
    i = i+1
  if ( i < k )
    A[i] = 1
}
```

Actual cost, Worst cost, Amortized cost = ?

Increment : Actual Cost

- Actual cost is linear in the number of bits flipped



0 0 0 0	Actual cost
0 0 0 1	1
0 0 1 0	2
0 0 1 1	1
0 1 0 0	3
0 1 0 1	1
0 1 1 0	2

Increment : Worst Cost

- *A*[0..*k*-1] : *k* bits
- Worst when all bits are flipped $\Theta(k)$



1 1 1 1
0 0 0 0

- *m* `Increments` on an initially zero k-bit counter takes time O(*mk*)

Increment : Aggregate Method

- Tighten the worst-case cost of *m* `Increments`
- $A[0]$ flips every time : m times
- $A[1]$ flips every other time : $m/2$ times
- $A[2]$ flips every fourth time : $m/4$ times ...
- $A[i]$ flips every fourth time : $m/2^i$ times
- The total number of flips in *m* `Increments` is

$$\sum_{i=0}^{\lfloor \lg m \rfloor} \frac{m}{2^i} < m \sum_{i=0}^{\infty} \frac{1}{2^i} = 2m = O(m)$$

- Amortized cost = $O(m) / m = O(1)$

Increment : Accounting Method

- Assign 2-baht amortized cost for an Increment
- 1 baht for flipping a 0-bit to 1-bit
- 1 baht kept at the 1-bit for later flipping back to 0

Increment : Accounting Method

Σ (amortized cost)

0

0	0	0	0

0

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

0	0	0	1

1

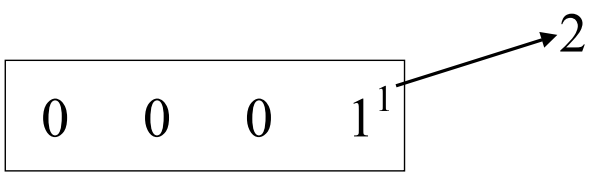
1

1^{st} `Increment`

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)



1

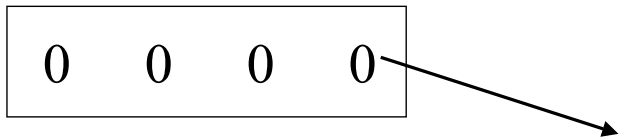
1st Increment

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

2+0



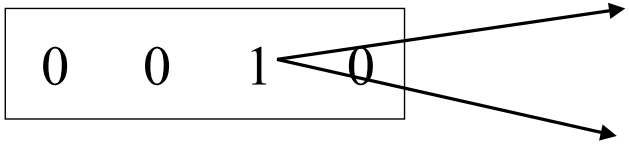
2nd Increment

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

2+1



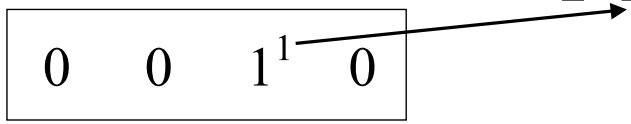
2nd Increment

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

2+2



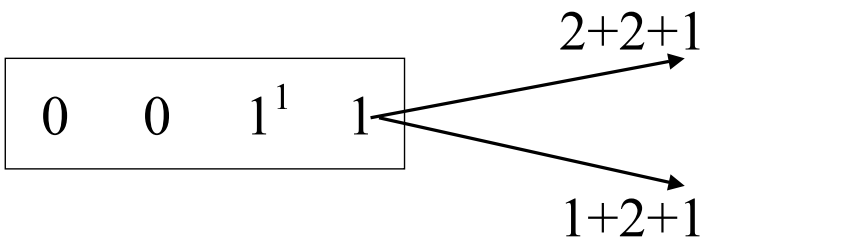
1+2

2nd Increment

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

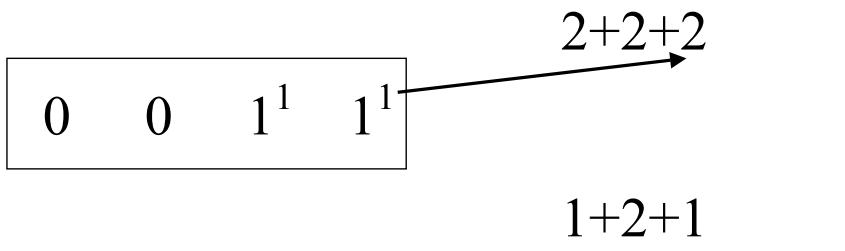


3rd Increment

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

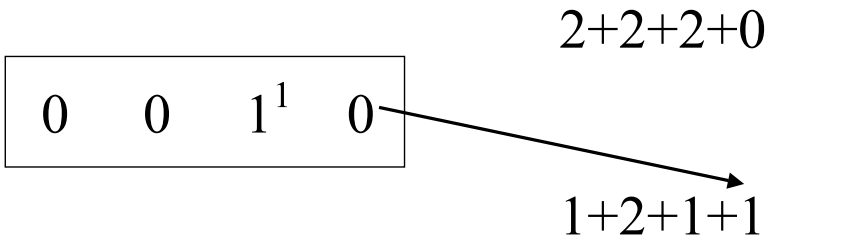


3rd Increment

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

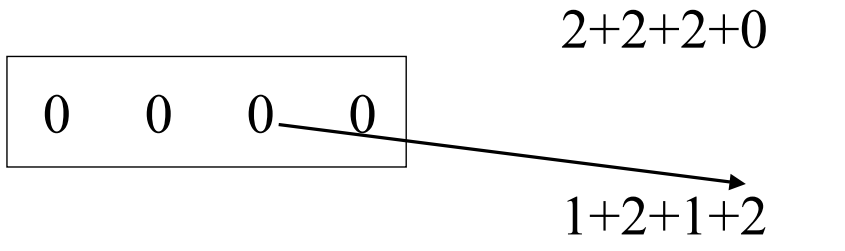


4th Increment

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

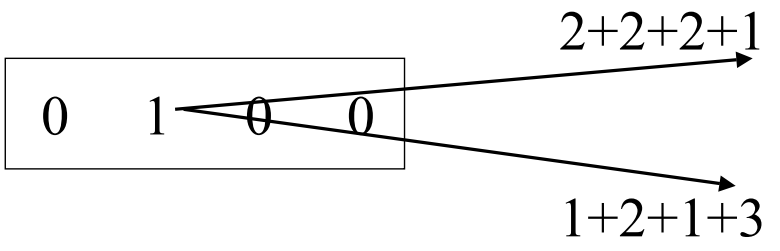


4th Increment

Σ (actual cost)

Increment : Accounting Method

Σ (amortized cost)

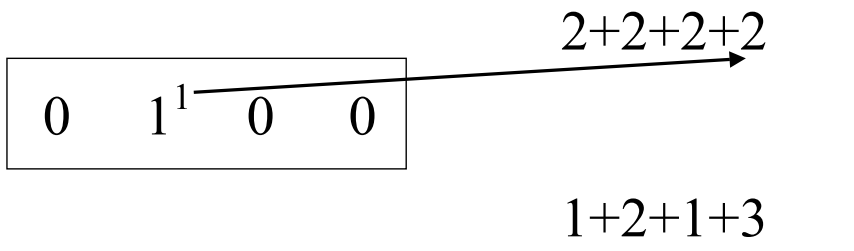


Σ (actual cost)

4th `Increment`

Increment : Accounting Method

Σ (amortized cost)



Σ (actual cost)

4th `Increment`

Increment : Accounting Method

Σ (amortized cost)

2+2+2+2

0 1 1 0 0

1+2+1+3

Σ (actual cost)

4th `Increment`

Increment : Accounting Method

- Each `Increment` costs at most two bahts since at most one bit is set to "1"
- No negative credit happens
- Σ(amortized cost) $\geq \Sigma$(actual cost) at all times
- `Increment` is O(1) amortized time

Increment : Potential Method

- Let Φ_i be potential function of the data structure after the i^{th} operation
- Let c_i be the actual cost of the i^{th} operation
- Let a_i be the amortized cost of the i^{th} operation

Potential Function

- Potential energy increases if $a_i > c_i$
- Potential energy decreases if $a_i < c_i$

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

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

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

$$= \sum c_i + \Phi_m - \Phi_0$$

$$\sum a_i \geq \sum c_i \quad \text{if } \Phi_m \geq \Phi_0$$

Increment : Potential Method

- Let b_i be the number of 1's in the counter after after the i^{th} `Increment`
- Suppose that the i^{th} operation resets r_i bits

$$c_i \leq r_i + 1$$

$$b_i \leq b_{i-1} - r_i + 1$$

0 1 0 1 1 $\quad b_{11} = 3$

0 1 1 0 0 $\quad b_{12} = b_{11} - r_{12} + 1$

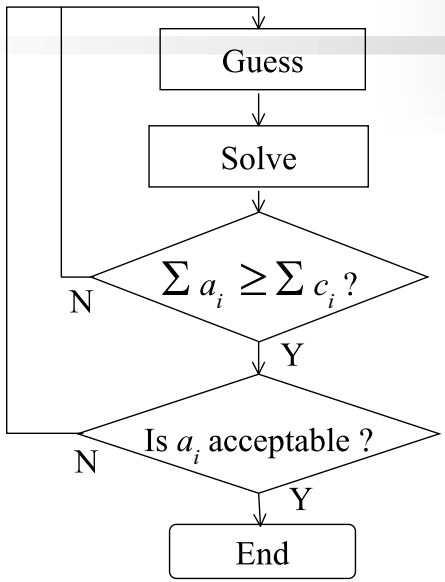
$$= 3 - 2 + 1 = 2$$

$$c_{12} = 3$$

Increment : Potential Method

- Let $\Phi_i = b_i$, $\Phi_0 = 0$, $\Phi_i \geq \Phi_0$
- $\Phi_i - \Phi_{i-1} = b_i - b_{i-1}$
- $\leq (b_{i-1} - r_i + 1) - b_{i-1}$
- $= 1 - r_i$
- $a_i = c_i + \Phi_i - \Phi_{i-1}$
- $\leq (r_i + 1) + 1 - r_i$
- $= 2 = O(1)$

Analysis Loop



Dy namic Table

- Table that can expand or contract as needed

```
Table_Insert( T, x )
  if ( T.size = 0 )
    T = CreateTable( 1 )
  if ( T.num = T.size )
    S = CreateTable( 2*T.size )
    insert all items of T into S   <-- O(n)
    freeTable( T )
    T = S
  insert x into Table[T]           <-- O(n)
```


Dynamic Table : Sequence of Inserts

- A sequence of n `Table_Insert` on initially empty table
- Double the table size when inserting into the full table
- Worst case of `Table_Insert` is O(n)
- Worst case of the sequence is O(n^2)
- Not tight : table expansion occurs infrequently

Worst Case of n Insertions

- Aggregate Method
- Observe that $c_i = i$ if i-1 is an exact power of 2, otherwise $c_i = 1$
- Amortized cost is 3 = O(1)

$$\begin{aligned}\sum_{i=1}^{n} c_i &\leq n + \sum_{j=0}^{\lfloor \lg n \rfloor} 2^j \\ &\leq n + 2^{\lfloor \lg n \rfloor + 1} - 1 \\ &< n + 2n \\ &= 3n\end{aligned}$$

Dynamic Table : Accounting Method

- Why is amortized cost of a `Table_Insert` 3 ?
- 1 is to move the item into the table (actual cost)
- 1 is kept for future movement during expansion
- 1 is kept for another older item in the table for future movement during expansion

Dynamic Table : Accounting Method

Σ (amortized cost)

0

0

Σ (actual cost)

Dynamic Table : Accounting Method

new item

Σ (amortized cost)

0

0

Σ (actual cost)

Dynamic Table : Accounting Method

new item

Σ (amortized cost)

0

0

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)



1

1

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)



2

1

Σ (actual cost)

Dynamic Table : Accounting Method



new item



Σ (amortized cost)

2

1

Σ (actual cost)

Dynamic Table : Accounting Method



new item





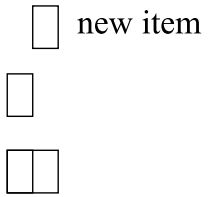
Σ (amortized cost)

2

1

Σ (actual cost)

Dynamic Table : Accounting Method



Σ (amortized cost)

2+0

1+1

Σ (actual cost)

Dynamic Table : Accounting Method



Σ (amortized cost)

2+1

1+2

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+2



1+2

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

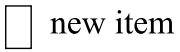
2+3



1+2

Σ (actual cost)

Dynamic Table : Accounting Method



Σ (amortized cost)

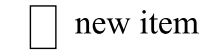
2+3



1+2

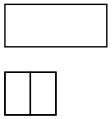
Σ (actual cost)

Dynamic Table : Accounting Method



Σ (amortized cost)

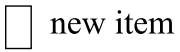
2+3



1+2

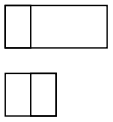
Σ (actual cost)

Dynamic Table : Accounting Method



Σ (amortized cost)

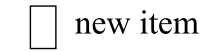
2+3+0



1+2+1

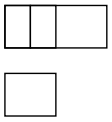
Σ (actual cost)

Dynamic Table : Accounting Method



Σ (amortized cost)

2+3+0



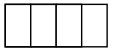
1+2+2

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+1



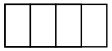
1+2+3

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+2



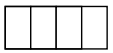
1+2+3

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+3



1+2+3

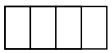
Σ (actual cost)

Dynamic Table : Accounting Method

new item

Σ (amortized cost)

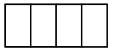
2+3+3



1+2+3

Σ (actual cost)

Dynamic Table : Accounting Method



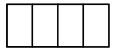
Σ (amortized cost)

2+3+3+1

1+2+3+1

Σ (actual cost)

Dynamic Table : Accounting Method



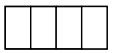
Σ (amortized cost)

2+3+3+2

1+2+3+1

Σ (actual cost)

Dynamic Table : Accounting Method



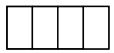
Σ (amortized cost)

2+3+3+3

1+2+3+1

Σ (actual cost)

Dynamic Table : Accounting Method



new item

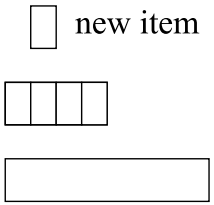
Σ (amortized cost)

2+3+3+3

1+2+3+1

Σ (actual cost)

Dynamic Table : Accounting Method



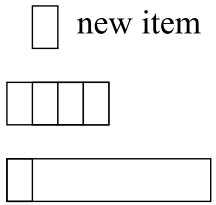
Σ (amortized cost)

2+3+3+3

1+2+3+1

Σ (actual cost)

Dynamic Table : Accounting Method



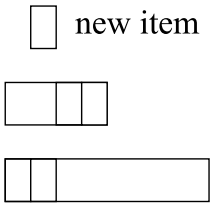
Σ (amortized cost)

2+3+3+3+0

1+2+3+1+1

Σ (actual cost)

Dynamic Table : Accounting Method



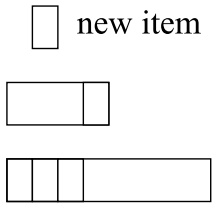
Σ (amortized cost)

2+3+3+3+0

1+2+3+1+2

Σ (actual cost)

Dynamic Table : Accounting Method



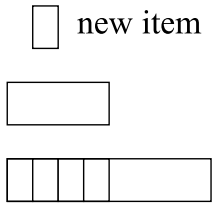
Σ (amortized cost)

2+3+3+3+0

1+2+3+1+3

Σ (actual cost)

Dynamic Table : Accounting Method



Σ (amortized cost)

2+3+3+3+0

1+2+3+1+4

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+3+3+1

1+2+3+1+5

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+3+3+2

1+2+3+1+5

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+3+3+3

1+2+3+1+5

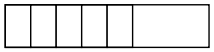
Σ (actual cost)

Dynamic Table : Accounting Method

new item

Σ (amortized cost)

2+3+3+3+3



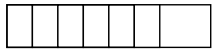
1+2+3+1+5

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+3+3+3+1



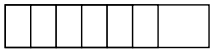
1+2+3+1+5+1

Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+3+3+3+2



1+2+3+1+5+1

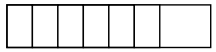
Σ (actual cost)

Dynamic Table : Accounting Method

Σ (amortized cost)

2+3+3+3+3+3

≥



1+2+3+1+5+1

Σ (actual cost)

amortized cost ≤ 3 = O(1)

Dynamic Table : Potential Method

- Let n_i be the number of items in the table
- Let s_i be size of the table
- We want the potential to be zero initially and after every expansion
- We want the potential to build to the table size when the table is full
- Potential is used for moving items during expansion

Dynamic Table :Potential Function

- $\Phi_i = 2n_i - s_i \qquad \Phi_i = 0$ when $i = 0$ or $n_i = s_i / 2$
- If the i^{th} insertion does not trigger an expansion
- $a_i = c_i + \Phi_i - \Phi_{i-1}$
- $= 1 + (2n_i - s_i) - (2n_{i-1} - s_{i-1})$
- $= 1 + (2n_i - s_i) - (2(n_i-1) - s_i)$
- $= 3$

Dynamic Table :Potential Function

- $\Phi_i = 2n_i - s_i \qquad \Phi_i = 0$ when $i = 0$ or $n_i = s_i / 2$
- If the i^{th} insertion does trigger an expansion ($n_{i-1} = s_{i-1}$)
- $a_i = c_i + \Phi_i - \Phi_{i-1}$
- $= n_i + (2n_i - s_i) - (2n_{i-1} - s_{i-1})$
- $= n_i + (2n_i - 2s_{i-1}) - (2s_{i-1} - s_{i-1})$
- $= 3n_i - 3s_{i-1}$
- $= 3n_i - 3(n_i-1)$
- $= 3$

Dynamic Table : Contraction

- Idea : halve the size of the table when deletion causes the table to become less than 1/4 full.
- See the CLR book for further details

Conclusion

- Amortized cost is the worst cost of sequence of operations
- Better estimate than m * (worst cost per operation)
- Used in the analysis of advanced data structures
- Accounting : guess $a_i \rightarrow$ no negative credit
- Potential function : guess $\Phi_i \rightarrow \Phi_i \geq \Phi_i \rightarrow a_i$