3. Amortized Analysis

Pukar Karki
Assistant Professor

Amortized Analysis

- In an **amortized analysis,** the time required to perform a sequence of data structure operations is averaged over all the operations performed.

Amortized Analysis

- While certain operations for a given algorithm may have a significant cost in resources, other operations may not be as costly.
- The **amortized analysis** considers both the costly and less costly operations together over the whole series of operations of the algorithm.

Amortized Analysis

- We can use **amortized analysis** to show that the average cost of operation is small, if one averages over a sequence of operations, even though a single operation within the sequence might be expensive.

Amortized Analysis

Three techniques:

- Aggregate analysis
- Accounting method
- Potential method

Concept of Aggregate Analysis

- ✔ In aggregate analysis for all n, a sequence of n operations takes worst case time T(n) in total. In the worst case the average cost or amortized cost per operations.

Cost/Operation = T(n)/n

Concept of Aggregate Analysis - Example

Example: We analyze stacks that have been augmented with a new operation.

- We have the two fundamental stack operations, each of which takes **O(1)** time.
- **PUSH(S, x)** pushes object **x** onto stack **S**.
- **POP(S)** pops the top of stack **S** and returns the popped object.

Concept of Aggregate Analysis - Example

- Since each of these operations runs in **O(1)** time, let us consider the cost of each to be 1.
- The total cost of a sequence of **n PUSH** and **POP** operations is therefore **n**, and the actual running time for n operations is therefore **Θ(n)**.

Concept of Aggregate Analysis - Example

- We add the stack operation **MULTIPOP(S, k)**, which removes the **k** top objects of stack **S**, or pops the entire stack if it contains less than **k** objects.

```
MULTIPOP(S, k)
1  while not STACK-EMPTY(S) and k ≠ 0
2       do POP(S)
3          k ← k - 1
```

Concept of Aggregate Analysis - Example

- What is the running time of **MULTIPOP(S, k)** on a stack of s objects?
- The number of iterations of the while loop is the number **min(s, k)** of objects popped off the stack.
- Thus, the total cost of **MULTIPOP** is **min(s, k)**, and the actual running time is a linear function of this cost.

```
MULTIPOP(S,k)
1  while not STACK-EMPTY(S) and k ≠ 0
2      do POP(S)
3         k ← k - 1
```

For each iteration of the loop, one call is made to POP in line 2.

Concept of Aggregate Analysis - Example

- Let us analyze a sequence of **n PUSH**, **POP**, and **MULTIPOP** operations on an initially empty stack.
- The worst-case cost of a **MULTIPOP** operation in the sequence is **O(n)**, since the stack size is at most **n**.

Concept of Aggregate Analysis - Example

- The worst-case time of any stack operation is therefore **O(n)**, and hence a sequence of **n** operations costs **O(n^2)**, since we may have **O(n) MULTIPOP** operations costing **O(n)** each.
- Although this analysis is correct, the **O(n^2)** result, obtained by considering the worst-case cost of each operation individually, is not tight.

Concept of Aggregate Analysis - Example

- Using the aggregate method of amortized analysis, we can obtain a better upper bound that considers the entire sequence of n operations.
- In fact, although a single **MULTIPOP** operation can be expensive, any sequence of **n PUSH**, **POP**, and **MULTIPOP** operations on an initially empty stack can cost at most **O(n)**.
- Why?

Concept of Aggregate Analysis - Example

- Each object can be popped at most once for each time it is pushed.
- Therefore, the number of times that **POP** can be called on a nonempty stack, including calls within **MULTIPOP**, is at most the number of PUSH operations, which is at most n.
- For any value of **n**, any sequence of **n PUSH, POP**, and **MULTIPOP** operations takes a total of **O(n)** time.
- The amortized cost of an operation is the average: **O(n)/n** = **O(1)**.

Accounting Method

- In the accounting method of amortized analysis, we assign differing charges to different operations, with some operations charged more or less than they actually cost.
- We call the amount we charge an operation its **amortized cost.**
- When an operation's **amortized cost** exceeds its **actual cost**, we assign the difference to specific objects in the data structure as **credit.**

Accounting Method

- Credit can help pay for later operations whose amortized cost is less than their actual cost.
- Thus, we can view the amortized cost of an operation as being split between its actual cost and credit that is either deposited or used up.
- Different operations may have different amortized costs.
- This method **differs** from **aggregate analysis**, in which all operations have the same amortized cost.

Accounting Method

- If we denote the actual cost of the **i^{th}** operation by **c_i** and the amortized cost of the **i^{th}** operation by **$\hat{c}_i$**, we require

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

for all sequences of n operations.

Accounting Method

- The total credit stored in the data structure is the difference between the total amortized cost and the total actual cost, or

$$\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i$$

- We must take care that the total credit in the data structure never becomes negative.

Accounting Method - Example

- Let us return to the stack example.
- Recall that the actual costs of the operations were

  PUSH = 1,

  POP = 1,

  MULTIPOP = min(k, s),

  where k is the argument supplied to MULTIPOP and s is the stack size when it is called.

Accounting Method - Example

- Let us assign the following amortized costs:

  PUSH = 2,

  POP = 0,

  MULTIPOP = 0.

- Here, all three amortized costs are constant.
- In general, the amortized costs of the operations under consideration may differ from each other, and they may even differ asymptotically.

Accounting Method - Example

- Suppose we use a dollar bill to represent each unit of cost.
- We start with an empty stack
- When we push something(suppose plate) on the stack, we use 1 dollar to pay the actual cost of the push and are left with a credit of 1 dollar (out of the 2 dollars charged), which we leave on top of the plate.
- At any point in time, every plate on the stack has a dollar of credit on it.

Accounting Method - Example

- The dollar stored on the plate serves as prepayment for the cost of popping it from the stack.
- When we execute a POP operation, we charge the operation nothing and pay its actual cost using the credit stored in the stack.
- To pop a plate, we take the dollar of credit off the plate and use it to pay the actual cost of the operation.
- Thus, by charging the PUSH operation a little bit more, we can charge the POP operation nothing.

Accounting Method - Example

- Moreover, we can also charge MULTIPOP operations nothing.
- To pop the first plate, we take the dollar of credit off the plate and use it to pay the actual cost of a POP operation.
- To pop a second plate, we again have a dollar of credit on the plate to pay for the POP operation, and so on.
- Thus, we have always charged enough up front to pay for MULTIPOP operations.

Accounting Method - Example

- In other words, since each plate on the stack has 1 dollar of credit on it, and the stack always has a non-negative number of plates, we have ensured that the amount of credit is always non-negative
- Thus, for any sequence of n PUSH, POP, and MULTIPOP operations, the total amortized cost is an upper bound on the total actual cost.
- Since the total amortized cost is O(n), so is the total actual cost.

Potential Method

- Instead of representing prepaid work as credit stored with specific objects in the data structure, the potential method of amortized analysis represents the prepaid work as “potential energy,” or just “potential,” which can be released to pay for future operations.

Potential Method

- We associate the potential with the data structure as a whole rather than with specific objects within the data structure.
- The potential method works as follows:

Potential Method

- We will perform n operations, starting with an initial data structure D_0.
- For each i = 1, 2, ... n, we let c_i be the actual cost of the i^{th} operation and D_i be the data structure that results after applying the i^{th} operation to data structure D_{i-1}.
- A potential function ϕ maps each data structure D_i to a real number $\phi(D_i)$, which is the potential associated with data structure D_i.
- The amortized cost $\hat{\mathbf{c}}_\mathbf{i}$ of the i^{th} operation with respect to potential function ϕ is defined by

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) .$$

Potential Method

- The amortized cost of each operation is therefore its actual cost plus the change in potential due to the operation.
- By previous equation, the total amortized cost of the n operations is

$$\begin{aligned}\sum_{i=1}^{n} \hat{c}_i &= \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) \\ &= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \ .\end{aligned}$$

Potential Method

- If we can define a potential function ϕ such that $\phi(D_n) \geq \phi(D_o)$ then the total amortized cost $\sum_{i=1}^{n} \hat{c}_i$ gives an upper bound on the total actual cost $\sum_{i=1}^{n} c_i$

Potential Method - Example

- The total amortized cost of n operations with respect to ϕ therefore represents an upper bound on the actual cost.

Potential Method - Example

- Let us now compute the amortized costs of the various stack operations.
- If the i^{th} operation on a stack containing s objects is a PUSH operation, then the potential difference is

$$\begin{aligned}\Phi(D_i) - \Phi(D_{i-1}) &= (s+1) - s \\ &= 1 .\end{aligned}$$

- By equation , $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) .$

the amortized cost of this PUSH operation is

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= 1 + 1 \\ &= 2 .\end{aligned}$$

Potential Method - Example

- Suppose that the i^{th} operation on the stack is MULTIPOP(S, k), which causes k' = min(k, s) objects to be popped off the stack.
- The actual cost of the operation is k', and the potential difference is

$$\Phi(D_i) - \Phi(D_{i-1}) = -k' .$$

- Thus, the amortized cost of the MULTIPOP operation is

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= k' - k' \\ &= 0 . \end{aligned}$$

- Similarly, the amortized cost of an ordinary POP operation is 0.

Potential Method - Example

- The amortized cost of each of the three operations is O(1), and thus the total amortized cost of a sequence of n operations is O(n).
- Since we have already argued that

$$\Phi(D_i) \geq \Phi(D_0)$$

  the total amortized cost of n operations is an upper bound on the total actual cost.
- The worst-case cost of n operations is therefore O(n).

Review Questions

1) If the set of stack operations included a **MULTIPUSH** operation, which pushes k items onto the stack, would the O(1) bound on the amortized cost of stack operations continue to hold?
2) Write short notes on:

a) Aggregate analysis

b) Accounting method

c) Potential method