Lecture: Analysis of Algorithms (CS483 - 001)

Amarda Shehu

Spring 2017

Amortized Analysis

- Amortized analysis refers to a strategy for analyzing a sequence of operations to show that the *average cost per operation* is small even though a specific operation or few infrequent operations within the sequence may be expensive.
- The basic idea is that we are talking about scenarios where a few operations may be expensive, but their cost is "amortized" by other operations that are frequent and inexpensive.
- Note: even though we are taking averages, no probabilistic analysis is involved. There is no analysis of inputs.
- An amortized analysis guarantees the average performance of each operation in the worst case.

Types of Amortized Analyses

Common amortization arguments:

- Aggregate method
- Accounting method
- Potential method

The aggregate method is the simplest. However, it lacks the precision of the other two. The accounting and potential methods allow to associate a specific amortized cost with each operation. The potential method is often the most difficult to apply.

Aggregate Method

- Consider a dynamic table: whenever we run out of space when inserting an element in the table, we (1) double the table size, (2) copy elements from old table to new table, and (3) free the storage of the old table.
- Consider a sequence of n insertions. The worst-case time to execute one insertion is $\theta(n)$.
- What is the worst-case time for n insertions?

Worst-case Analysis with Aggregate Method

The worst-case cost for n insertions is $\theta(n) << \theta(n^2)$
Let c_i be the cost associated with the i^{th} insertion.

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

Then, the cost of n insertions is:

$$\begin{aligned} &= \sum_{i=1}^{n} c_i \\ &\leq n + \sum_{j=0}^{\lfloor lgn \rfloor} 2^j \\ &\leq 3n \\ &\in \theta(n). \end{aligned}$$

Hence, the average cost of an insertion in a dynamic table is $\theta(n)/n \in \theta(1)$.

Accounting Method

- Charge the i^{th} operation a fictitious amortized cost $\hat{c}_i$, where \$1 pays for 1 unit of work (time).
- This fee is consumed to perform the operation (debit). Any amount of work not immediately consumed is stored in the bank for use by subsequent operations (credit).
- The (total) bank balance must never go negative.
- We must ensure:

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

- The total amortized costs (right handside) provide a tight upper bound on the total true costs (left handside).

Accounting Analysis of Dynamic Tables

- Charge an amortized cost of $\hat{c}_i = \$3$ for the i^{th} insertion
- \$1 pays for the insertion itself
- \$2 is stored to be consumed by the eventual table doubling: \$1 pays to move itself upon table doubling; the other \$1 pays to move an item that has already moved once when table doubles in size
- Key invariant: the bank balance never goes negative
- Let's trace this over a few table doublings

Trace of Accounting Analysis of Dynamic Tables

Here is a table that tracks what goes on. True cost is written as 1 + (i-1), where i-1 is 0 for the cheapest operations.

Operation	True Cost	Credit or Debit
1	1	+2
2	1 + 1	+1
3	1 + 2	0
4	1	+2
5	1 + 4	-2
6	1	+2
7	1	+2
8	1	+2
9	1 + 8	-6

Continued

10	1	+2
11	1	+2
12	1	+2
13	1	+2
14	1	+2
15	1	+2
16	1	+2
17	1 + 16	-14
...	...	...

Observation: The number of light operations (that store credit) between heavy operations (that accumulate debit) increases exponentially (as power of two).

Accounting Analysis of Dynamic Tables

- Since each heavy operation costs 1 + (i-1), it will require a debit of 3 - [1 + (i-1)]. Who will pay for this? The operations above that have credit with them. While it may seem in the beginning that these credits are soon evaporated, if you continue the analysis to say 17 insertions, you will see that it becomes more rare to have a heavy operation, and the number of insertions that store credit increases (as powers of 2).
- Even something expensive down the road like operation 33 that will require 1 + 32 units of work, will have enough to cover its debit of 3 - 33 = -30, because operations from 18 to 32 (15 of them) have stored a total credit of 2 * 15 = 30, which is enough.
- So $3n$ is an upper bound on the total cost needed to pay for n insertions, which is another way to see that the cost of each insertion in a dynamic table is $\theta(1)$

Potential Method: Basic Idea

- Most powerful and formal method, but also the hardest to use.
- A generalization of the accounting method, useful in cases when it is difficult to assign credit to specific operations.
- **Basic Idea:** Treat the bank account as a potential energy of the dynamic table. Rather than associating a credit or debit with each operation (as in the accounting method), the potential method stores pre-payments as a potential or potential energy that can be spent to pay for later operations.

Potential Method: Basic Idea

- The potential energy $\phi(D)$ is a function of the entire data structure D.
- An operation i changes the potential energy of the data structure by increasing or decreasing it as the operation changes the state of the data structure.
- After operation i, the state of the data structure is D_i. Its energy is $\phi(D_i)$. The change in potential energy after operation i is $\phi(D_i) - \phi(D_{i-1})$. This is denoted as $\Delta\phi_i$.
- The total energy should never go negative, in the same way that the bank balance in the accounting method should never go negative.

Potential Method

Framework:

- Start with an initial data structure D_0
- Operation i transforms D_{i-1} to D_i
- The true cost of operation i is c_i
- Define a potential function $\phi : \{D_i\} \rightarrow \mathcal{R}$, such that $\phi(D_0) = 0$ and $\phi(D_i) \geq 0$ for all i
- The amortized cost $\hat{c}_i$ with respect to ϕ is defined to be $\hat{c}_i = c_i + \phi(D_i) - \phi(D_{i-1})$

Inspiration for Potential Method Comes from Physics

The amortized cost of an operation i is:

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

In physics, $\phi(D_i) - \phi(D_{i-1})$ is the potential difference $\Delta\phi_i$

- If $\Delta\phi_i > 0$, then $\hat{c}_i > c_i$. This means that operation i stores work/credit in the data structure for later use
- If $\Delta\phi_i < 0$, then $\hat{c}_i < c_i$. This means that the data structure delivers up stored work/energy to help pay for operation i

Amortized Costs Bound the True Costs

Is it true that:

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

Let's see. The total amortized cost of n operations is:

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

since $\phi(D_n) \geq 0$ and $\phi(D_0) = 0$.

Coming Up with Potentials

- Our task is to find a function ϕ s.t. $\phi_0 = 0$ and $\phi_i \geq 0$ for all i.
- If we do this, then we have proved that the total actual/true cost will be no greater than the total amortized cost ($\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i$).
- How does one come up with potentials ϕ? There are no general rules. This is what makes applications of the potential method difficult for amortized analysis.
- Trial and error is the best approach.
- What are good potentials? Those that do not overestimate the total true cost by too much (that is, total amortized cost is a tight upper bound of total true cost).

A Potential for Dynamic Table

Let n be the number of elements in the table and m the table size. When the table gets full, i.e., $n > m$, then the n elements are moved into a table of size $2m$.

Now let $\phi(D_i) = 2n_i - m_i$ be the potential of the dynamic table after the i^{th} insertion.

Questions:

- Why is $\phi(D_0) = 0$
- Why is $\phi(D_i) \geq 0$ for all i
- Why is $\frac{1}{n} * \sum_{i=1}^{n} \hat{c}_i \in \theta(1)$
- What is $\hat{c}_i$ when table stays the same and when table doubles?

Potential for Dynamic Table: Basic Properties

Let $\phi(D_i) = 2n_i - m_i$ after the i^{th} insertion.

- $\phi(D_0) = 2 \cdot n_0 - m_0 = 2 \cdot 0 - 0 = 0$
- $\phi(D_i) = 2 \cdot n_i - m_i$
  1. If no doubling: $n_i = n_{i-1} + 1$, $m_i = m_{i-1} = 2 \cdot (n_{i-k})$. Since $n_i \geq n_{i-k}$, $2 \cdot n_i \geq m_i$. Hence, $\phi(D_i) \geq 0$.
  2. Else: $n_i = n_{i-1} + 1$, $m_i = 2 \cdot m_{i-1}$, $n_{i-1} = m_{i-1}$. Hence, $2 \cdot n_i - m_i = 2 \cdot (n_{i-1} + 1) - 2 \cdot n_{i-1} = 2 \geq 0$

Potential for Dynamic Table: Amortized Costs

Let $\phi(D_i) = 2n_i - m_i$ after the i^{th} insertion.

- $\Delta\phi_i = (2 \cdot n_i - m_i) - (2 \cdot n_{i-1} - m_{-1})$
- $\hat{c}_i = c_i + \Delta\phi_i$
  1. If no doubling: $n_i = n_{i-1} + 1$, $m_i = m_{i-1}$, and $c_i = 1$. Hence, $\Delta\phi_i = 2 \cdot (n_i - n_{i-1}) - (m_i - m_{-1}) = 2 \cdot 1 + 0 = 2$. Therefore, $\hat{c}_i = c_i + \Delta\phi_i = 1 + 2 = 3$.
  2. Else: $n_i = n_{i-1} + 1$, $m_i = 2 \cdot m_{i-1}$, $n_{i-1} = m_{i-1}$, and $c_i = n_i$. Hence, $\Delta\phi_i = 2 \cdot (n_i - n_{i-1}) - (m_i - m_{-1}) = 2 \cdot 1 - 2 \cdot m_{i-1} + m_{i-1} = 2 - m_{i-1}$. Therefore, $\hat{c}_i = c_i + \Delta\phi_i = n_i + 2 - m_{i-1} = n_i + 2 - n_{i-1} = (n_i - n_{i-1}) + 2 = 1 + 2 = 3$.
- Hence $\sum_{i=1}^{n} \hat{c}_i = 3 \cdot n$. It follows that $\sum_{i=1}^{n} c_i \leq 3 \cdot n$. The average cost per operation is $\frac{1}{n} \cdot 3n = 3 \in \theta(1)$.

Another Potential for Dynamic Table

Let $\phi(D_i) = 2i - 2^{\lceil (lgi)+1 \rceil} + 1$ be the potential of the dynamic table after the i^{th} insertion.

Questions:

- Why is $\phi(D_0) = 0$
- Why is $\phi(D_i) \geq 0$ for all i
- What is $\Delta\phi_i$?
- What is $\hat{c}_i$?
- Why is $\frac{1}{n} * \sum_{i=1}^{n} \hat{c}_i \in \theta(1)$

Conclusion on A Good Potential Function

- Different potential functions lead to different amortized time bounds.
- Difficulty is to get the best possible potential function.
- A good potential function goes up a little during any cheap/fast operation and goes down a lot during any expensive/slow operation.
- Unfortunately, there are no general techniques for doing this other than trying lots of possibilities, guided by experience and domain-specific insight.

Conclusion on Amortized Analysis

- Amortized costs provide a nice abstraction on the cost performance of operations on a data-structure.
- Any of the analysis methods can be used.
- A method may be simpler or more precise than the others in different scenarios.
- Different schemes may work for assigning amortized costs in the accounting method.
- Different potentials may work in the potential method.