Amortized analysis: The accounting method

Let us use push() and multi-pop() operations for stacks as an example. The following is a sample of formal written arguments and also what you are expected to do in homework. Remember, we always start with an empty data-structure (e.g., an empty stack in this context).

Assign the deposit amount to each operation:

- $2 for each push() operation.
- $0 for each multi-pop() operation.

The key is then to prove that the bank balance remains non-negative during a series of push() and multi-pop() operations. There are multiple ways that one can argue about this claim. One particular way is to establish the following invariant during the execution of a series of operations.

\[ \text{# (bank) credits} = \text{# items in the stack} \]

For our purpose, it suffices to prove the \( \geq \) direction. It happens to be tight so that we have \( = \). One can also phrase the above invariant in terms of proof by inductions. The induction hypothesis could be

After the \( i \)th operation, \( \text{# (bank) credits} = \text{# items in the stack} \).

Note that the above invariant (or induction hypothesis) implies that the balance is non-negative after each operation because the number of items in the stack is non-negative.

Let us continue with the induction language which is strict in mathematics. First, we argue about the base case \( i = 0 \), where no operation has been performed. The bank starts with 0 balance and the stack is empty. Thus, the induction hypothesis holds.

Now, we know after the \( i \)th operation, the bank balance \( c \) equals the number of items \( k \) in the stack. We want to establish the induction hypothesis after the \( (i+1) \)th operation. If the \( (i+1) \)th operation is

- push(), then we deposit $2 and spend one for the push operation. The new bank balance \( c' = c + 1 \) and the new number of items \( k' = k + 1 \). Thus \( c' = k' \) after the \( (i+1) \)th operation.

- multi-pop, then we deposit $0 and spend \( k \) for the multi-pop operation. The new bank balance \( c' = c - k \) and the new number of items \( k' = k - k \). Because \( c = k \), we have \( c' = k' \) after the \( (i+1) \)th operation.

Thus, we complete the proof by induction. The induction hypothesis implies that the bank balance is non-negative so that we are done. The amortized complexity of push() and multi-pop() is hence \( O(1) \).

Amortized analysis: The potential function method

We continue with the push and multi-pop example, but with the potential function method. As we have shown in the lecture, it suffices to find the right potential function \( \Phi \) such that \( \Phi_0 = 0 \) and \( \Phi_i \geq 0, \forall i \).

1Contents will appear in homework and exams.
In our example, we choose $\Phi_i$ to be the number of items in stack after the $i$th operation. It is easy to verify that $\Phi_0 = 0$ and $\Phi_i \geq 0, \forall i \geq 1$.

With the definition of the potential function, one can easily derive the amortized complexity of each operation. Thus, the crucial part of the potential function method is to find an appropriate potential function.

- **Push()**: $t' = t_i + \Phi_i - \Phi_{i-1} = 1 + 1 = 2$, which is $O(1)$.
  The change in the potential is an increase in one, which combines with the constant-time operations of push to yield a total amortized cost of 2.

- **Multi-pop()**: Multipop(k): $t' = t_i + \Phi_i - \Phi_{i-1} = k + -k = 0$, which is $O(1)$.
  The potential decrease cancels the running time of multi-pop().