
1: Amortized Analysis

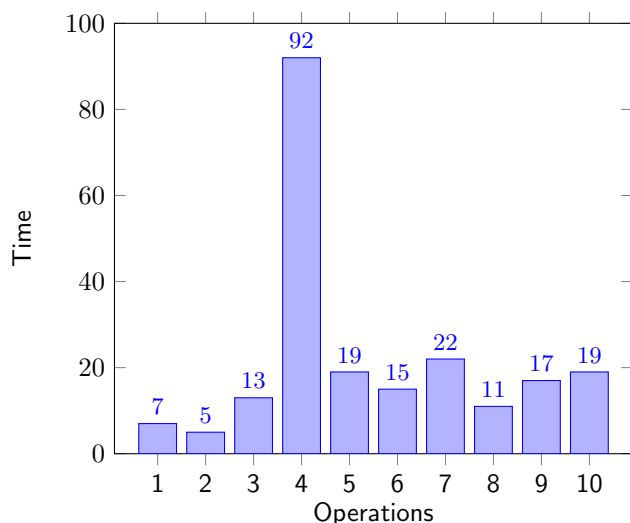
1. True or False:

- (a) Amortized analysis considers the worst-case cost of an operation over a sequence of operations.
- (b) If an operation takes $O(n)$ worst-case time, it is possible for it to have $O(1)$ amortized time.
- (c) If an operation takes $O(1)$ expected time, then it takes $O(1)$ amortized time.

2. The aggregate method of amortized analysis can be thought of simply summing the costs of an entire sequence of operations and dividing by the number of operations to obtain an average cost per operation.

The accounting method of amortized analysis is slightly more complex, and can be thought of as "how much resources (ie: money) are needed per step in order to sustain all future operations?". You can think of being able to store resources that were not used to pay for operations as "credit", with the goal of ensuring the credit never becomes negative. The goal then becomes trying to find the smallest number of resources to give at each step that still leaves enough credit to pay for more expensive operations.

Given a series of ten operations below, what is the amortized cost per operation using the aggregate method? What is the amortized cost per operation using the accounting method?

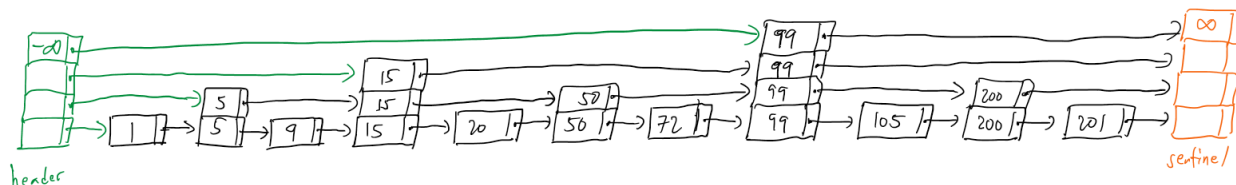


3. Suppose we have a simple stack data structure with PUSH, POP, and COPY operations. This stack has a max size of k ; after every k PUSH and POP operations, the stack is copied. We already know that PUSH and POP are $O(1)$ and that COPY is $O(n)$ for n number of elements. Using the accounting method, how much should you charge per operation to ensure the amount of credit never goes negative using the stack? Show that the cost of n stack operations is $O(n)$.

2: Skip Lists

1. For every perfect skip list, there is a nearly "equivalent" binary search tree. Discuss what we might mean by equivalent.

For the skip list below from lecture, construct an "equivalent" binary search tree.



Does this work for a randomized skip list? Why or why not?

2. In the construction of a Skip List, we promoted nodes to the next higher layer with probability $\frac{1}{2}$, or stopped promoting with probability $\frac{1}{2}$.

What if we promote nodes with probability $\frac{1}{3}$ instead, and stop promoting with probability $\frac{2}{3}$? Does this $\frac{1}{3}$ -Skip List still have $O(\log n)$ search time? Justify your answer.

3. What if at each layer j , we use a different promotion probability? We will always promote to layer 1. However, for any layer $j > 1$, we will promote with probability $\frac{j-1}{j}$, and stop promoting with probability $\frac{1}{j}$ (in layer 2, the promotion probability is $\frac{1}{2}$; in layer 3, promotion probability is $\frac{2}{3}$; in layer 4, promotion probability is $\frac{3}{4}$, and so on)? Let's call this a tall Skip List because we stop promoting with a lower probability, so the number of layers is greater. Does this tall Skip List still have $O(\log n)$ search time? Justify your answer. If not, what runtime will it have?

3: Solutions

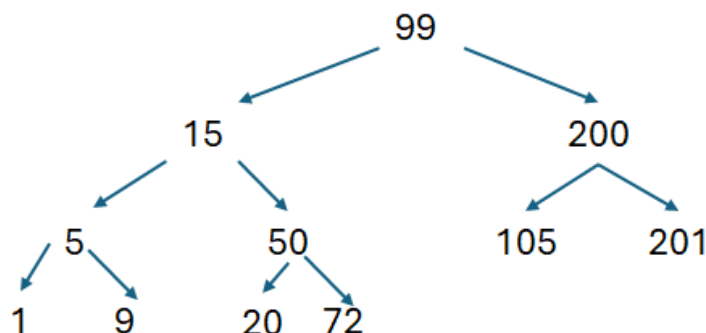
Amortized Analysis

- (a) False; amortized analysis considers the average cost per operation over a sequence of operations.
 - (b) True; an operation with a high worst-case time can still have a low amortized time over a series of operations.
 - (c) False; while expected time is a probabilistic assessment of the average time an operation takes, amortized time is a deterministic measure. $O(1)$ expected time does not guarantee the same bound when considering the worst-case distribution of operations.
- Using the aggregate method, the sum of all the costs of operations divided by the number of operations becomes $220/10 = 22$.
Using the accounting method, the smallest amount we can allocate to the first three operations such that enough credit remains to pay for the expensive fourth operation is 30.
- Using the accounting method, we would need to charge \$2 for every PUSH and POP operation, and \$0 for each COPY operation. This is because during the construction of the k elements in a stack, we must "stockpile" money to eventually pay for a COPY operation, which will require \$ k dollars (one dollar for each of the k elements in the stack). Thus, prior to every COPY operation, we need to pay for every PUSH and POP operation (\$1 each) and stockpile also \$1 for each PUSH/POP for the eventual COPY operation.

Since the "cost" per operation remains constant, we know the amortized cost per operation is $O(1)$. Over a stack of n operations, this would equate to a total cost of $O(n)$.

Skip list

- By equivalent, we mean that the pattern of left-right movement in the BST is almost the same as the pattern of down-right movement in the perfect skip-list. Note that it may not be an exact translation because there may be edge effects when the number of items isn't a power of 2.



This may not work for a randomized skip list. Remember that a randomized skip list could just be a linked list without any promotion, as unlikely as that is.

2. Yes, it's still $O(\log n)$ search. Changing the probability to $1/3$ changes the base of the logarithm from 2 to 3 for the number of layers, but that is only a constant order change. Note that for a regular skip list, you would spend on average 2 steps on each layer, but in this skip list, you now spend on average 3 steps per layer, but that is still a constant, so still $O(\log n)$.
3. No. In fact, it is likely $\Omega(n)$. Notice that the tall skip list is $O(n)$ layers tall in expectation, so clearly going up/down the skip list will take more than logarithmic time.