

15-451/651 Algorithm Design & Analysis, Fall 2026

Recitation #4

Objectives

- Practice defining potential functions and performing amortized analysis
- Practice using union-find data structure

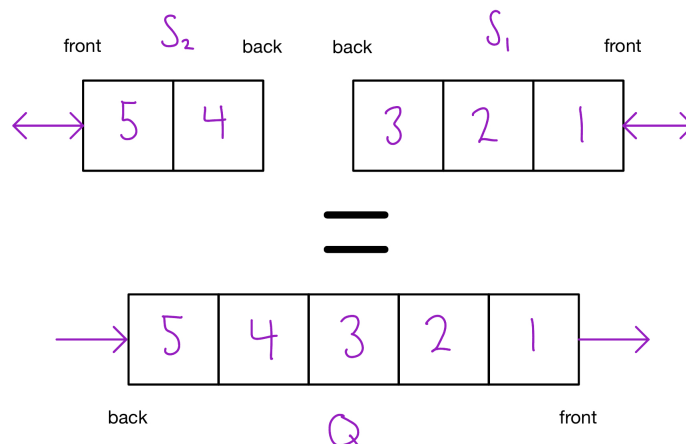
Recitation Problems

1. **(Deque)** There is a classic method to construct a first-in-first-out queue with $O(1)$ amortized cost operations `pushBack` and `popFront` from two last-in-first-out stacks with cost 1 operations `pushFront`, `popFront`, and `size` as follows:

Let stacks S_1 and S_2 represent the front/head and back/tail of the queue Q respectively. Then implement the operations by moving all elements from S_2 to S_1 whenever we would need to remove but it is empty:

```
pushBack(x) {  
    S2.pushFront(x)  
}  
  
popFront() {  
    if (S1.size() == 0) {  
        for i in range(S2.size()) {  
            S1.pushFront(S2.popFront())  
        }  
    }  
    return S1.popFront()  
}
```

Under this implementation the amortized bounds follow from setting $\Phi(S_1, S_2) = 2|S_2|$ and correctness follows from the invariant that the elements of Q are always equal to the elements of S_1 appended to the elements of S_2 in reverse.



One natural extension of this is to implement a deque, which in addition to the standard queue functionality, also provides `pushFront` and `popBack`, allowing users to insert and remove from either side as they please.

- (a) Suppose you provide symmetric implementations of those methods as follows:

```
pushFront(x) {
    S1.pushFront(x)
}

popBack() {
    if (S2.size() == 0) {
        for i in range(S1.size()) {
            S2.pushFront(S1.popFront())
        }
    }
    return S2.popFront()
}
```

Using the aggregate method, give a sequence of n operations under which each operation has $\Omega(n)$ amortized cost.

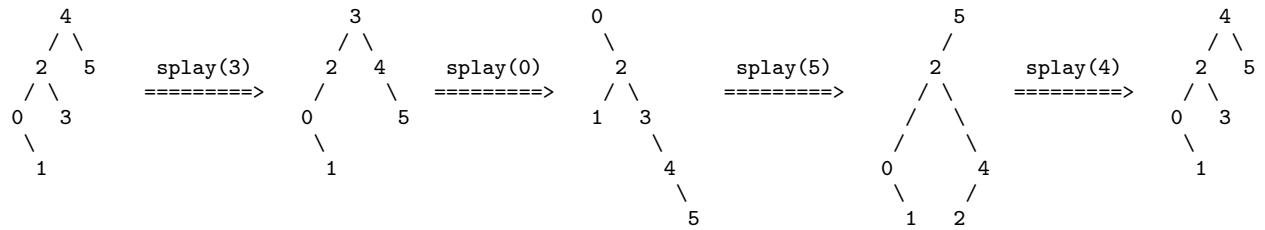
- (b) Now suppose you had access to a third stack S_3 to use for temporary processing. Come up with a way to implement the deque operations that maintains the invariant but avoids the expensive case above.

- (c) Define a potential function $\Phi(S_1, S_2)$ and use it to prove that the operations you defined have $O(1)$ amortized cost.

- (d) Suppose that you start with an empty deque and then perform n operations (push or pop from either the front or the back). Bound the total actual cost of these operations.

2. **(Cyclic Splaying)** Starting from a tree T_0 of n nodes a sequence of $\ell \geq 1$ splay operations

is done. It turns out that the initial tree T_0 and the final tree T_ℓ are the same. Let k be the number of *distinct* nodes splayed in this sequence. (Clearly $k \leq \ell$.) Below is an example where $k = \ell = 4$ and $n = 6$.



It's easy to see that the average number of splaying done per splay at most $1 + 3 \log_2 n$. How?

It's a bit more difficult to prove that you can bound this by $1 + 3 \log_2 k$. Show how to do this.
Hint: You'll have to make use of a non-uniform weight assignment.