04-L

栈与队列

双栈当队

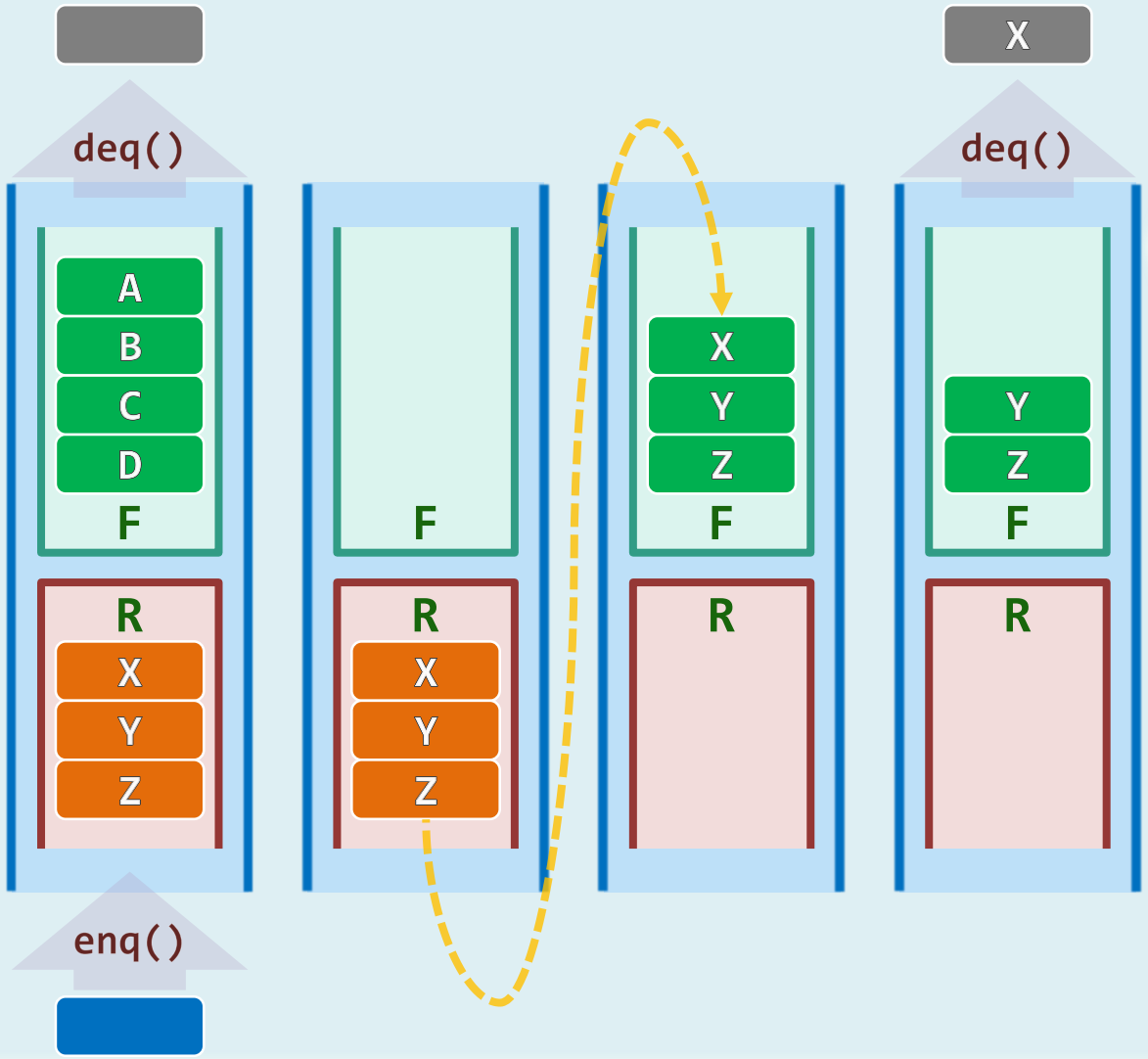
不恒其德，或承之羞，贞吝

我们也不用当场付款，要了什么东西都由店家记在一个小账本上，每两星期结一次账。

邓俊辉

deng@tsinghua.edu.cn

Queue = Stack x 2



- `def Q.enqueue(e)`

```
    R.push(e);
```

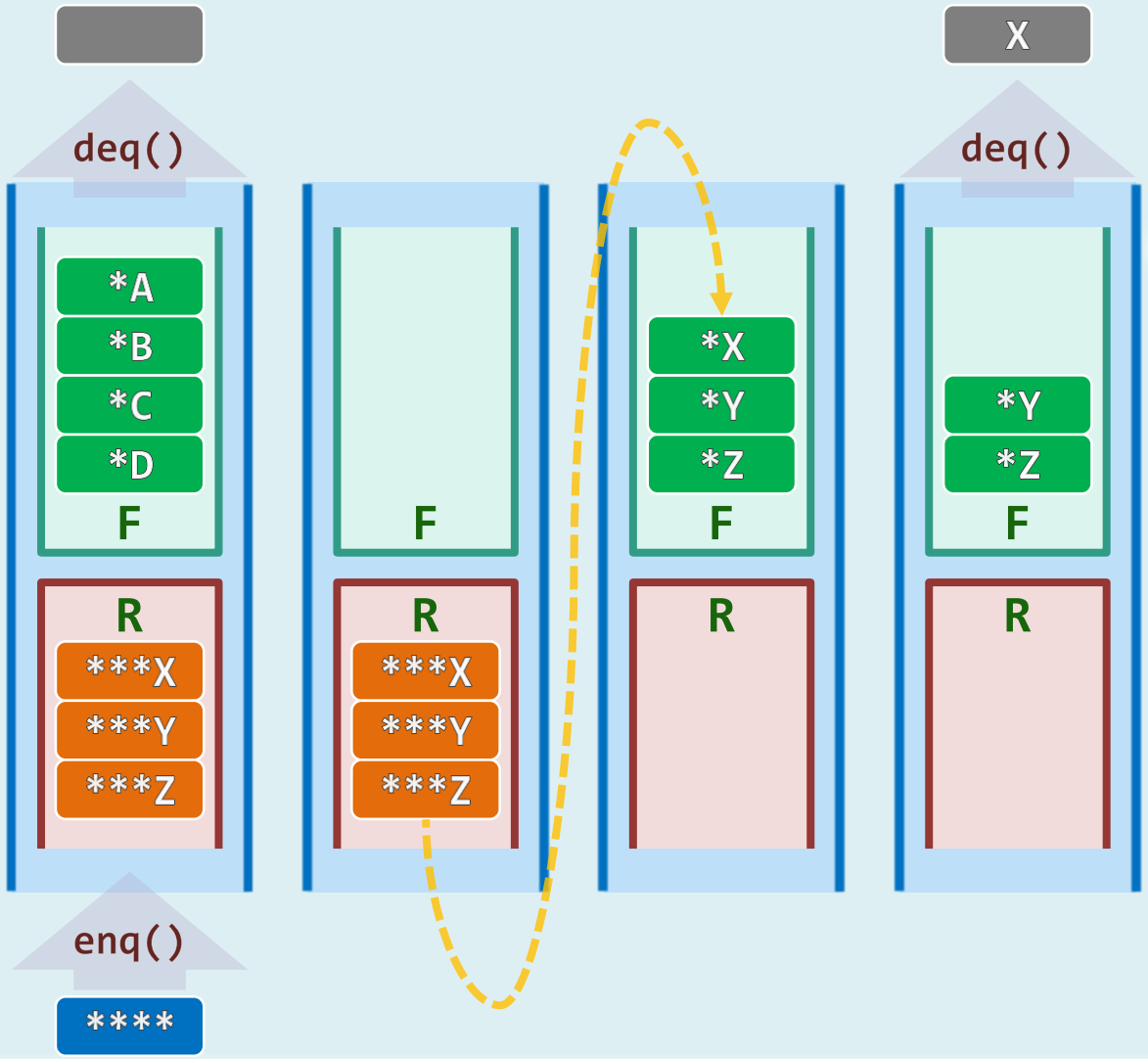
- `def Q.dequeue() # 0 < Q.size()`

```
    if ( F.empty() )
        while ( !R.empty() )
            F.push( R.pop() );
    return F.pop();
```

- Best/worst case: $O(1)/O(n)$

  Average? Amortization!

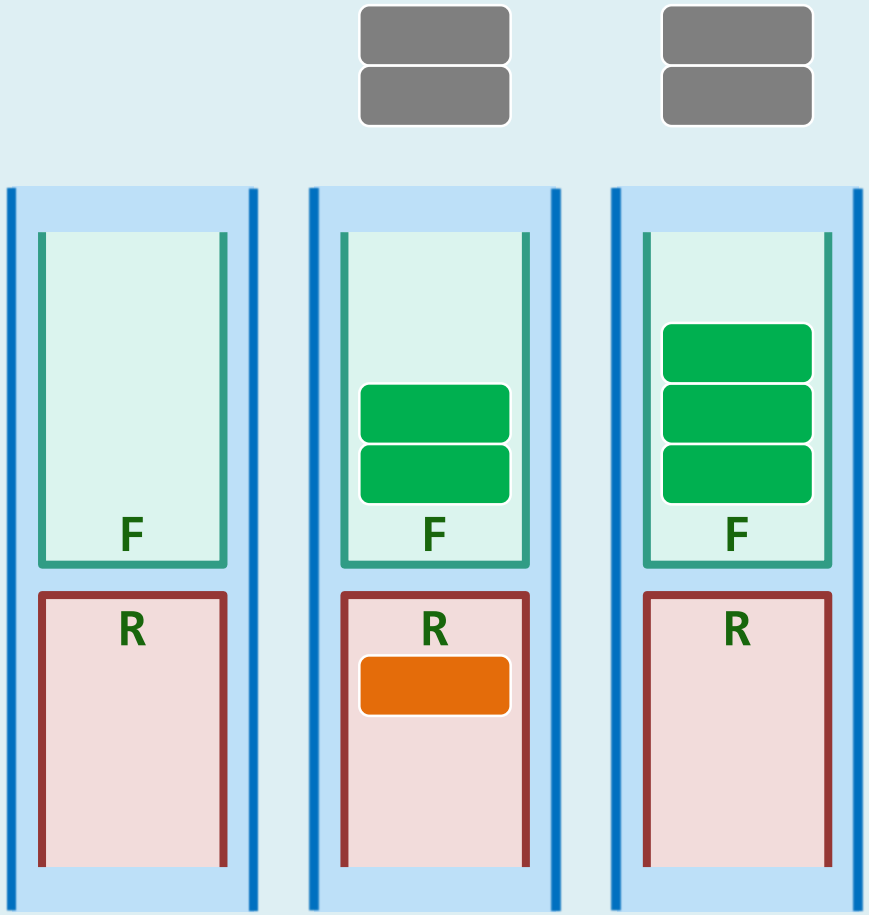
Amortization By Accounting



- Assign each new element with 4 coins //deposit
  - 1 for its enqueue()
  - 2 for transfer, and
  - the last 1 for dequeue()
- Hence every operation is pre-paid and ...
- The structure will never run out of credit
- Amortized cost of any operation sequence involving n ITEMs is

  $4n = O(n)$

Amortization By Aggregate



- Consider the moment when

  d dequeue()'s and **e** enqueue()'s

  have been done //d <= e

- The time cost

  for **ALL** the operations is

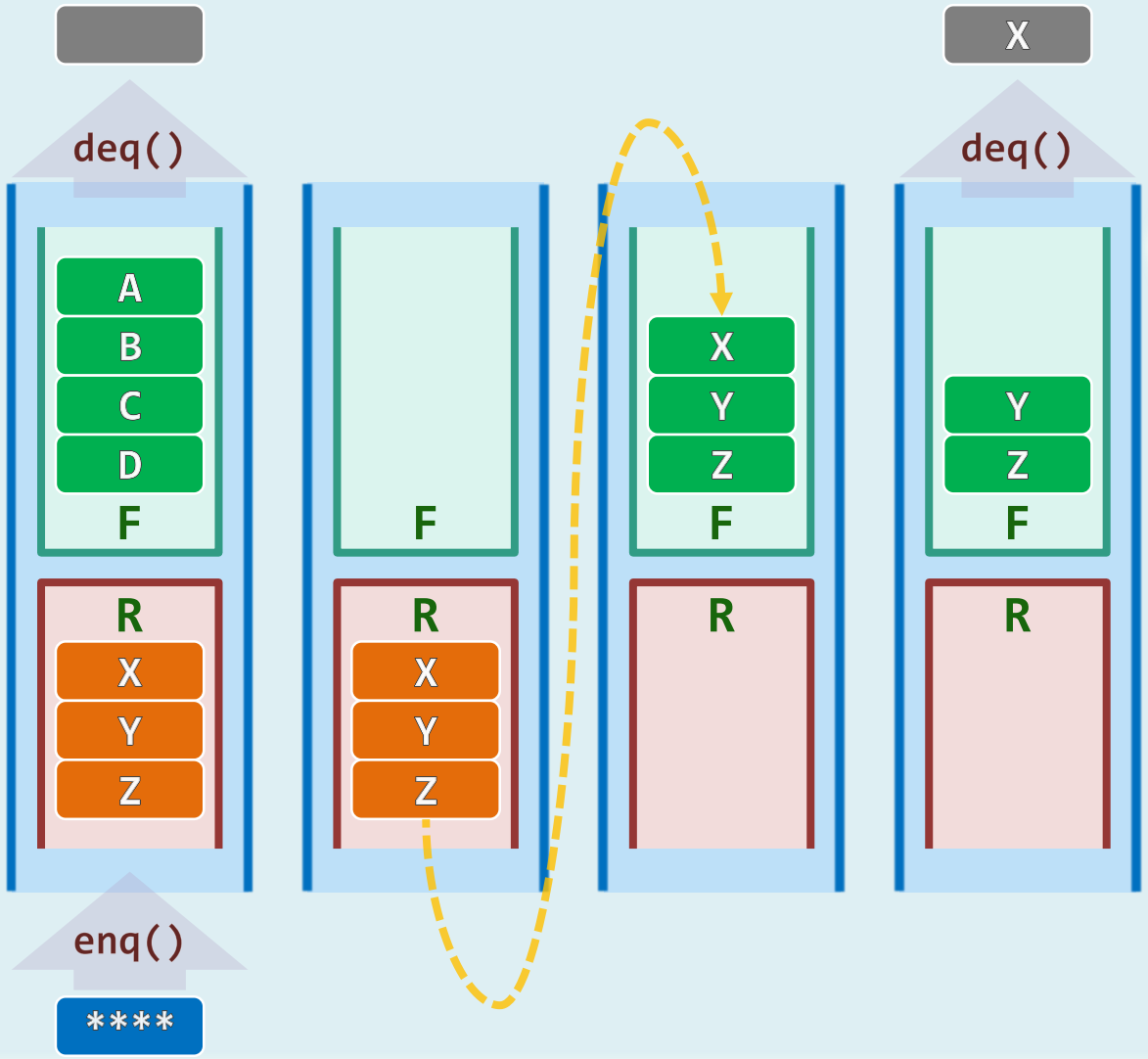
  $$\leq 4 \cdot d + 3 \cdot (e - d) = 3e + d$$

- The amortized cost

  for each OPERATION is

  $$\frac{3e + d}{e + d} < 3$$

Amortization By Potential



- Consider the k^{th} operation
- Define $\Phi_k = |R_k| - |F_k|$
- Then $A_k = T_k + \Phi_k - \Phi_{k-1} \equiv 2$
- Hence

$$2n \equiv \sum_{k=1}^{n} A_k = \sum_{k=1}^{n} T_k + \Phi_n - \Phi_0$$

$$2n = T(n) + \Phi_n - \Phi_0 > T(n) - n$$

$$T(n) < 3n = \mathcal{O}(n)$$