Aufgabe 5.5

Mittwoch, 7. Juni 2017 08:14

a)

```
public int dequeue() {
  if(s2.isEmpty())
    while(!s1.isEmpty()
      s2.push(s1.pop())
  return s2.pop();
}
```

O(1) O(1) + O(n) O(1) O(1)

O(1) + O(1) + O(n) + O(1) + O(1)
=> O(n)

Die Laufzeit für dequeue beträgt O(n)

b)
Tatsächliche Laufzeiten:

"Teure" Operation, wenn s2 ist leer
sonst "billige" Operationen:

O(1)
O(1) + n * c (c steht für den Kopiervorgang)

Laufzeiten fürs Amortisationschema:

T(enqueue): c

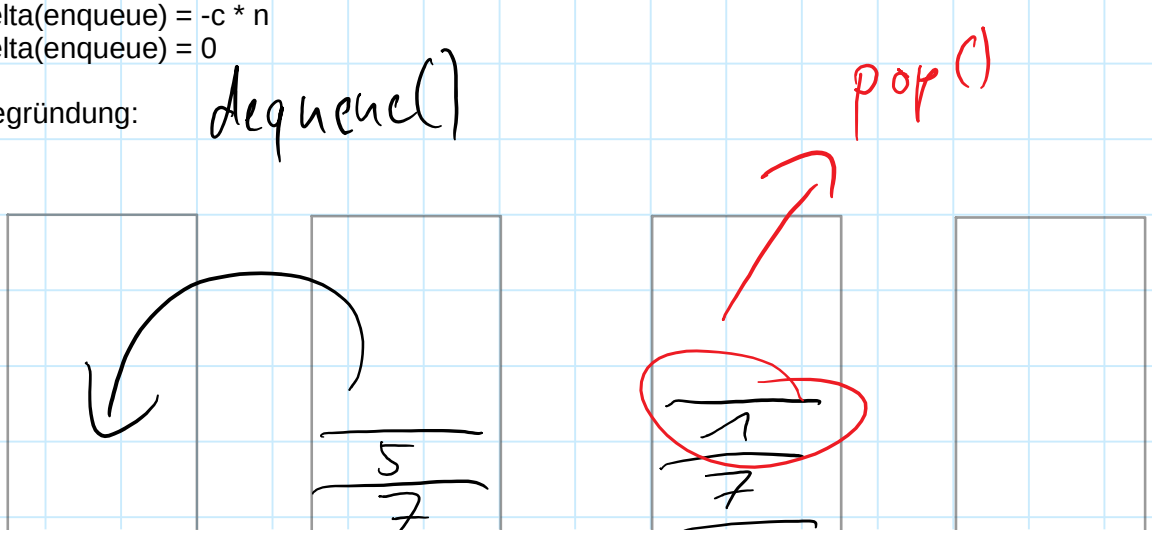
T(dequeue):
"Teure" Operation, wenn s2 ist leer
sonst "billige" Operationen:

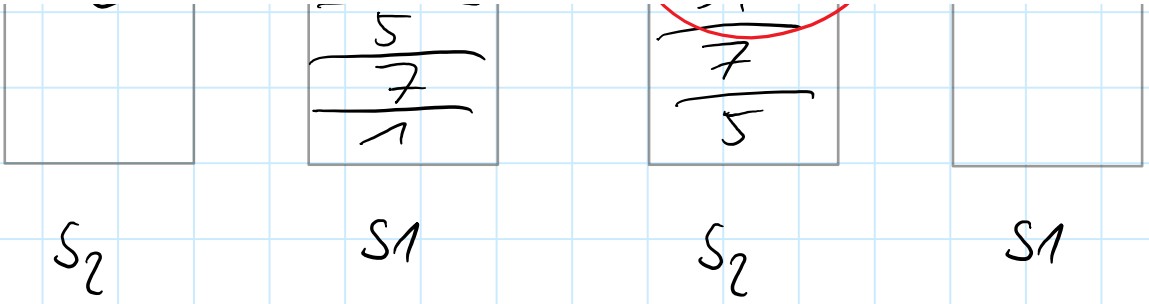
wenn s2 leer: n * -c
wenn s2 nicht leer: 0

Das Tokenkonto wird nie leer, da

delta(enqueue) = -c * n
delta(enqueue) = 0

Begründung: dequeue()





Das Tokenkonto ist nie negativ:

- Bei dem Aufruf von dequeue() und wenn s2 negativ ist, müssen alle Elemente von s1 in s2 kopiert werden, was die Anzahl der Elemente mal der Laufzeit kostet, die Kopieren kostet.
- Aber so viele Elemente wurden bereits auf den Stack von s1 durch billige Operationen gespeichert und somit passt das Schema

q.e.d.

A(enqueue) = T(enqueue) + $\Delta(\text{enqueue}) = O(1)$

A(dequeue) = T(dequeue) + $\Delta(\text{enqueue}) = O(1) + c * n - c * n$ = O(1)