# Blatt 4

## Exercise 1

The queue must persist of two dynamic arrays:

One for input and one for output.

### ENQUEUE

Enqueue operations are pretty straight-forward, because when we assume to push to push an arbitrary number n+1 to an input-array of the size n, then the last push (n+1) takes O(n) time, while all the others take constant time.

So, to amortize it, we can simply average the taken time (n+1) by the time it takes to perform the size doubling (n):

$$O((n+1)/n) = O((n/n) + (1/n)) = O(1)$$

### DEQUEUE

Dequeueing is quite more complicated. We must differentiate two scenarios:

output-array is not empty:
we just have to pop one element, which runs in constant time O(1)

output-array is empty:
dequeueing takes O(n) time, because we have to move all the elements from the input array to the output array.

so, if we combine the scenarios, we know that after moving n elements from the input to the output, we can perform n dequeue-operations in constant time, before the output array is empty again.

So, we have a amortized execution time of O(n/n) = O(1), making it a constant amortized.

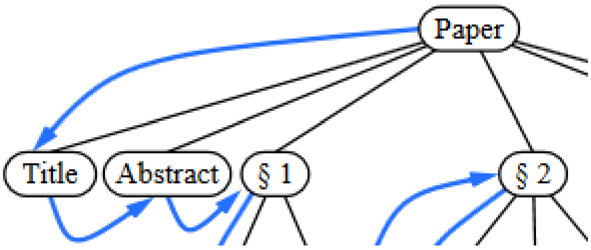
## Exercise 2

See other paper with code :^}

## Exercise 3

Note: I actually didn't fully understand what "Return the position visited after p" means.

For me, it means to return the position of a node that follows the given p - in one of the specific order.




So if i pass Abstract to the preordered traversal method, it will return the position of § 1.

See the other paper with some code :^}