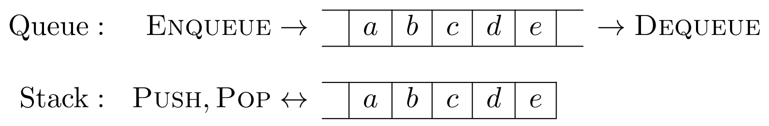
Mid-term examination on Prolog

Christian Rinderknecht

20 October 2008

A *queue* is a like stack where items are pushed on one end and popped on the other end. Adding an item in a queue is called *enqueuing*, whereas removing one is called *dequeuing*. Compare the following figures:

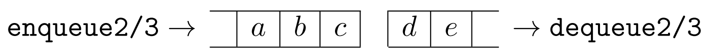


If `enqueue(I,In,Out)` is proved, then `Out` is the queue `In` in which item `I` has been enqueued; if `dequeue(In,Out,I)` is proved, then `Out` is the queue `In` from which item `I` has been dequeued.

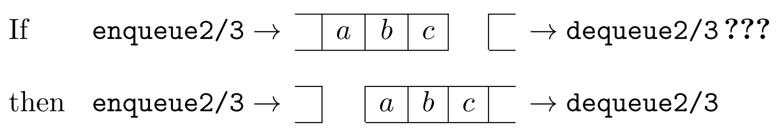
Questions. Define predicates `enqueue1/3`, `dequeue1/3`, `enqueue2/3` and `dequeue2/3` for the two following cases.

1. **A one-stack implementation.** A simple idea to implement one queue is to use one stack. In this case, `enqueue1/3` is simply pushing and `dequeue1/3` removes the item at the bottom of the stack.

2. **A two-stack implementation.** We can implement a queue with two stacks instead of one: one for enqueuing, one for dequeuing.

   So `enqueue2/3` is pushing on the first stack and `dequeue2/3` is popping on the second. If the second stack is empty, we swap the stacks and reverse the (new) second:

   Let the pair {`S,T`} denote the queue where `S` is the stack for enqueuing and `T` the stack for dequeuing.

Question. Which of the one-stack or two-stack implementation is the most efficient for dequeuing? What are the best and worst cases for each?