Implement Queue with a single Stack

October 24, 2011 allaboutalgorithms Go to comments Leave a comment

This is a very common problem that most of you might have even figured out. But I know a few who wouldn't know how to do it. What would you have done if you could have used 2 stacks? You would have been manipulating the two, popping and pushing elements between the two to just make them work like a queue. Basically, you would have been using double the required space.

> *With 1 stack, the only way (as you might have figured out) is to use recursion. Once this concept comes to mind, things become sweet and simple. For Enqueuing, you just need to PUSH an element onto the stack. So that is no hassle. For Dequeuing, what you need to do is recurse to the bottom of the stack by POP-ing elements (which is somewhat similar to how we used to tackle tree-related problems with recursion) and then print the last element. You again PUSH the rest of the elements on to the stack while coming bottom->up.*

Here is a pseudo code for both the functions :

```
void ENQUEUE ( element )
{
stacky_queue.PUSH (element) //stacky_queue is basically my stack
}
bool DEQUE ()
{
If ( NOT stacky_queue.EMPTY())
{
popped_element = stacky_queue.POP()
IF ( NOT DEQUE() )
{
PRINT popped_element
}
ELSE
{
stacky_queue.PUSH(popped_element)
}
return true
}
return false
}
```

Cool, eh? Hope you got it! If not read the paragraph in italics once more and you certainly will understand it then! 🙂