CS2002301 & EC2002302 Data Structures

Homework #1 Reference Solution (problem #1 only)

- Stack Solution

- Basic idea

Every time we push a number into the stack, we also record "the number of consecutive increments". If we find the number of consecutive increments == N, we pop the stack N times.

- Example

Suppose that $S = [4, 5, 1, 2, 3, 6, 7]$, $T = 3$:

Step 1. Push (4, 1) into the stack. The first element in the tuple represents the number in the sequence, the second represents the number of consecutive increments.

Step 2. Push (5, 2) into the stack. Since 5>4(the top element in stack), we know that the numbers have increased 1(recorded in the second element of the tuple) +1 times.

Step 3. Push (1, 1) into the stack.

Step 4. Push (2, 2) into the stack.

Step 5. Push (3, 3) into the stack.

Step 6. Because the second element in the top tuple == N, we pop the element N times. Now, the stack contains [(4, 1), (5, 2)]

Step 7. Push (6, 3) into the stack.

Step 8. Because the second element in the top tuple == N, we pop the element N times. Now, the stack is empty.

Step 9. Push (7, 1) into the stack.

Step 10. 7 is the end of the sequence. The program terminates here.

- Optimization suggestions

Well, if you use the above algorithm without any optimization, you can still pass all test cases (~70ms for public cases and ~900ms for hidden cases). However, there are still some ways to optimize your program:

1. Rewrite the pop() function of the stack. That is, you have to make sure that when you encounter a large amount of pop operations, you can still complete the operations in a short time.
2. Dynamic allocation.