Stack from queue

Method 1 (By making push operation costly)

This method makes sure that newly entered element is always at the front of ‘q1’, so that pop operation just dequeues from ‘q1’. ‘q2’ is used to put every new element at front of ‘q1’.

push(s, x) // x is the element to be pushed and s is stack

1) Enqueue x to q2

2) One by one dequeue everything from q1 and enqueue to q2.

3) Swap the names of q1 and q2

// Swapping of names is done to avoid one more movement of all elements

// from q2 to q1.

pop(s)

1. Dequeue an item from q1 and return it.

Method 2 (By making pop operation costly)

In push operation, the new element is always enqueued to q1. In pop() operation, if q2 is empty then all the elements except the last, are moved to q2. Finally the last element is dequeued from q1 and returned.

push(s, x)

1) Enqueue x to q1 (assuming size of q1 is unlimited).

pop(s)

1) One by one dequeue everything except the last element from q1 and enqueue to q2.

2) Dequeue the last item of q1, the dequeued item is result, store it.

3) Swap the names of q1 and q2

4) Return the item stored in step 2.

// Swapping of names is done to avoid one more movement of all elements

// from q2 to q1.

Queue using stacks

Method 1 (By making enQueue operation costly) This method makes sure that oldest entered element is always at the top of stack 1, so that deQueue operation just pops from stack1. To put the element at top of stack1, stack2 is used.

enQueue(q, x)

1) While stack1 is not empty, push everything from stack1 to stack2.

2) Push x to stack1 (assuming size of stacks is unlimited).

3) Push everything back to stack1.

Here time complexity will be O(n)

deQueue(q)

1) If stack1 is empty then error

2) Pop an item from stack1 and return it

Here time complexity will be O(1)

Method 2 (By making deQueue operation costly)In this method, in en-queue operation, the new element is entered at the top of stack1. In de-queue operation, if stack2 is empty then all the elements are moved to stack2 and finally top of stack2 is returned.

enQueue(q, x)

1) Push x to stack1 (assuming size of stacks is unlimited).

Here time complexity will be O(1)

deQueue(q)

1) If both stacks are empty then error.

2) If stack2 is empty

While stack1 is not empty, push everything from stack1 to stack2.

3) Pop the element from stack2 and return it.

Here time complexity will be O(n)

Method 2 is definitely better than method 1.

Method 1 moves all the elements twice in enQueue operation, while method 2 (in deQueue operation) moves the elements once and moves elements only if stack2 empty.

Implementation of method 2: