Computer Science & IT

Data Structure & programming

Queue

Lecture No. 03

By- Abhishek Sir

Recap of Previous Lecture

Topic

Topic

Topic

Topic

Topic

Circular queue

How to implement queue using Stack

Slide

Topics to be Covered

- Topic
- Topic
- Topic
- Topic
- Topic

Queue using stack.

Slide

Implementation of Queue Using Stack

Enqueue(1)

4
3
2
1

Implementation of Queue Using Stack

Enqueue(1)

= push(1)

Implementation of Queue Using Stack

Enqueue(1)

S_1

Slide

Implementation of Queue Using Stack

Enqueue(2)

Slide

Implementation of Queue Using Stack

Enqueue(2)

Slide

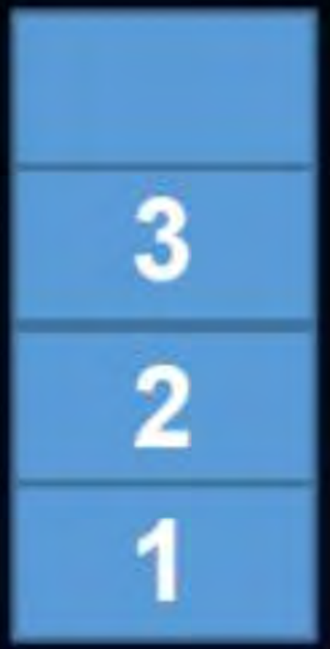
Implementation of Queue Using Stack

Enqueue(3)

Slide

Implementation of Queue Using Stack

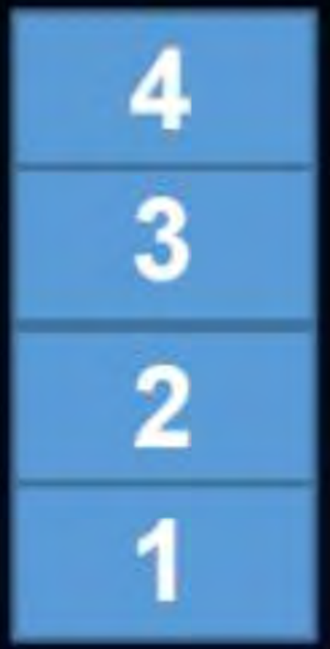
Enqueue(3)



Slide

Implementation of Queue Using Stack

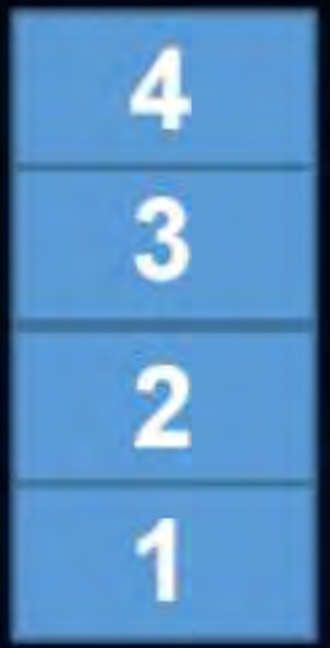
Enqueue(4)



Slide

Implementation of Queue Using Stack

Enqueue(4)



Stack

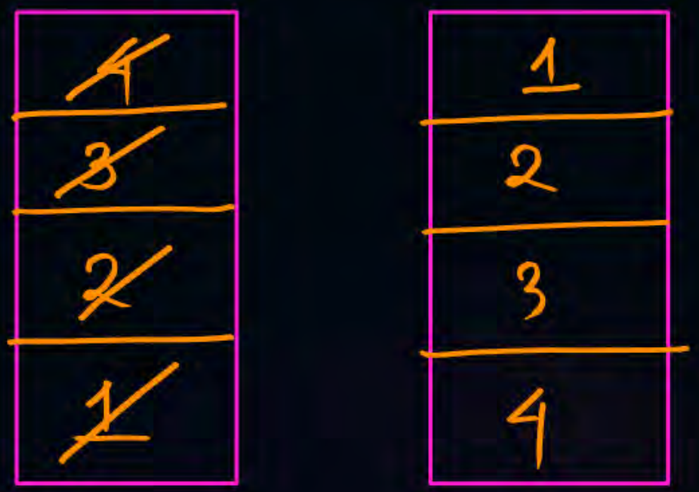
Dequeue should return 1.

S_1

Slide

Implementation of Queue Using Stack

1.

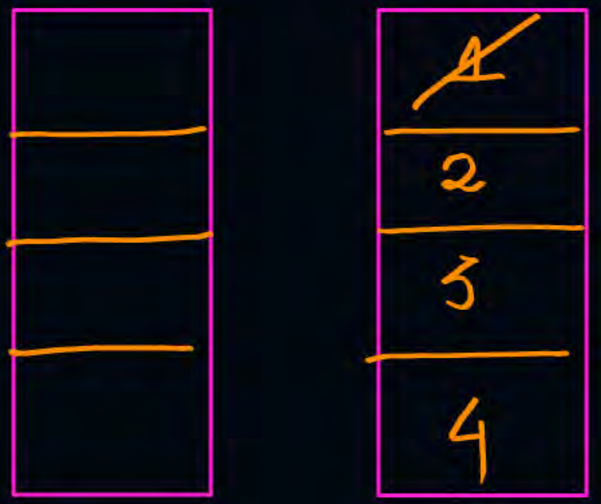


S_1 S_2 push

pop

until S_1 is empty

2.

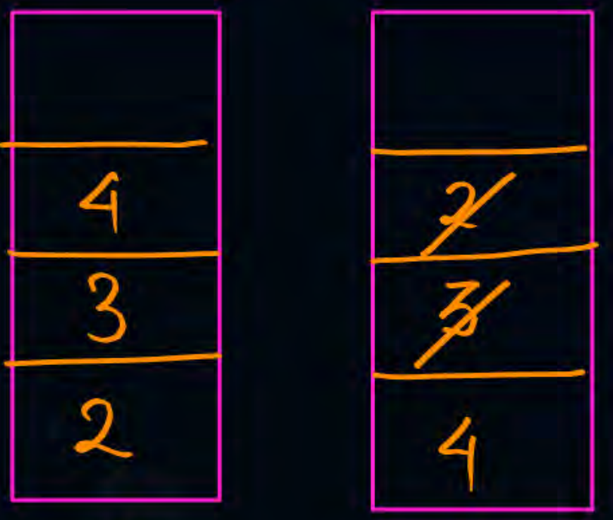


S_1 S_2

② pop S_2 ✓

return 1

3.

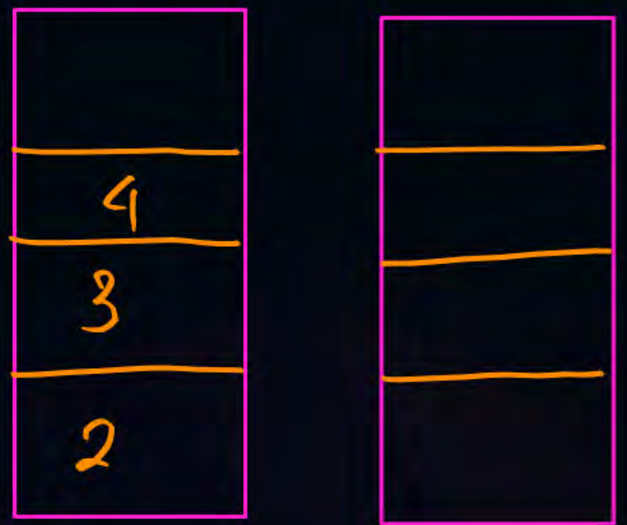


S_1 S_2

pop S_2 & push in S_1

Slide

Implementation of Queue Using Stack



S_1 S_2

final status of S_1, S_2

Slide

Implementation of Queue Using Stack

Linear

In the above implementation of queue if n Enqueue and 1 dequeue operation perfumed then total number of push and pop done is _____

n - Enqueue

1 - Dequeue

total

push

Enqueue n times

≡ push

n - push

pop

2n pop

2n-1 push

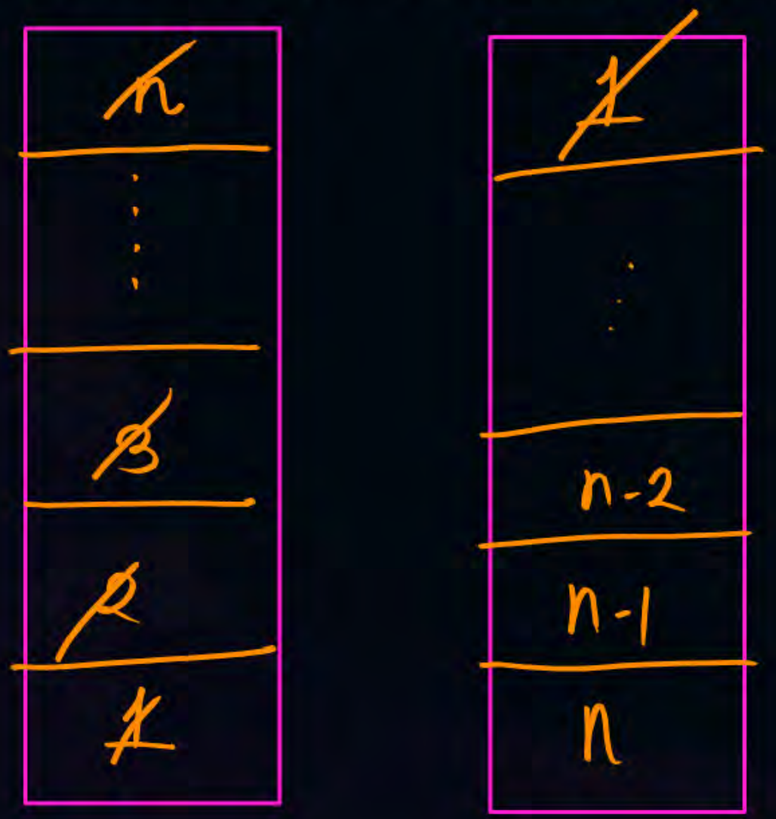
push: 3n-1

pop : 2n

Slide

Implementation of Queue Using Stack

In the above implementation of queue if n Enqueue and 1 dequeue operation perfumed then total number of push and pop done is _____

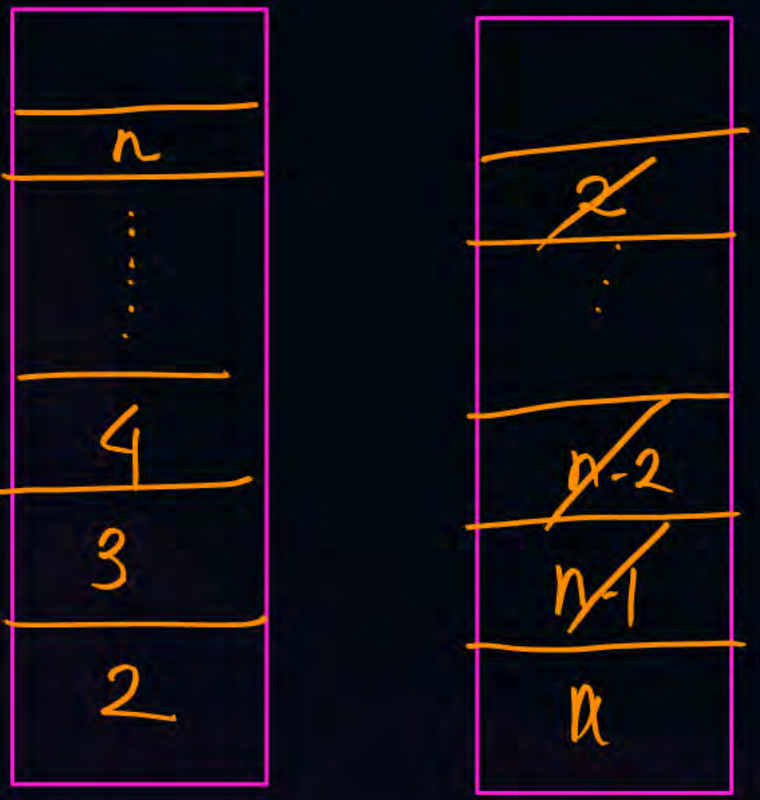


1. n pop
   n push
2. 1 pop
3.

Slide

Implementation of Queue Using Stack

In the above implementation of queue if n Enqueue and 1 dequeue operation perfumed then total number of push and pop done is _____



1. n pop S_1
   n push S_2
2. 1 pop — S_2
3. n-1 pop S_2
   n-1 push S_1

2n pop, 2n-1

Slide

Implementation of Queue Using Stack

In the above implementation of queue if n Enqueue and 1 dequeue operation perfumed then total number of push and pop done is _____

Implementation of Queue Using Stack

In the above implementation of queue if 50 Enqueue operation and 2 dequeue operation performed . If total number of push is x and total number pop is y then x+y is

50 Enqueue

50 push

2n pop ⟶ 2×50 = 100

2n-1 p ⟶ 99 push

dequeue

444 Ans

n = 49

2n - pop = 98

2n-1 push - 97

2 dequeue

Slide

Implementation of Queue Using Stack

In the above implementation of queue if 50 Enqueue operation and ~~10~~ 5 dequeue operation performed . If total number of push is x and total number pop is y then x+y is

_____

Slide

Implementation of Queue Using Stack

In the above implementation of queue if 50 Enqueue operation and ~~10~~ 5 dequeue operation performed . If total number of push is x and total number pop is y then x+y is

91 — 100

1000
45
955

50 Enqueue
= 50 push

1 deque	2nd deque	3rd deque	4th deque	5th deque
2n - pop = 100	n : 49	n : 48	n : 47	n : 46
2n - 1 push, 99	98 pop	96 pop	pop : 94	pop : 92
=	97 push	95 push	push : 93	push : 91

955 + 50 = 1005

Slide

Q. Suppose a stack implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE with respect to this modified stack?

(A) A queue cannot be implemented using this stack.

(B) A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes a sequence of two instructions.

(C) A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction.

(D) A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction.

(GATE_2014_2 M)

Enqueue ≡ push

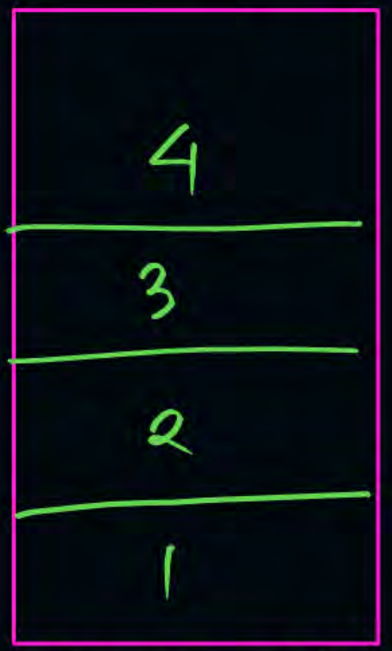
dequeue ≡ reverse
pop

Enqueue - Reverse
push
Reverse

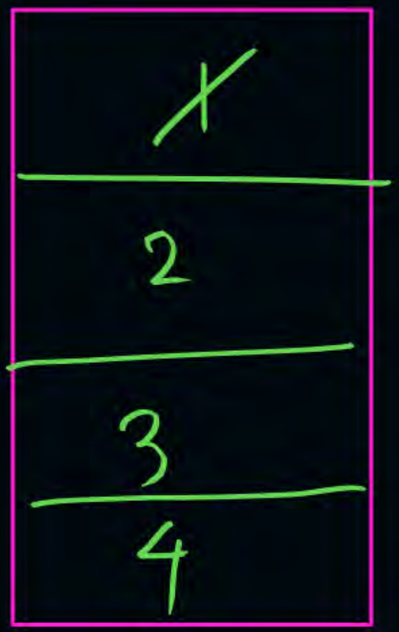
Dequeue = pop

① Enque(1)
② Enque(2)
③ Enque(3)
④ Enque(4)
⑤ dequeue ( )
⑥ Enque(5)
⑦ dequeue ( ) ← ②

Enqueue = push
dequeue - Reverse
~~push~~ pop

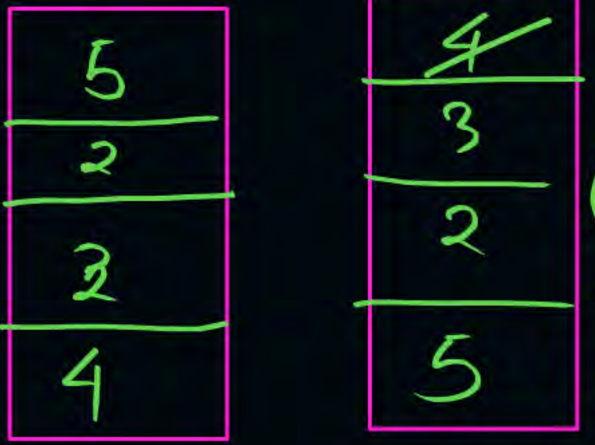


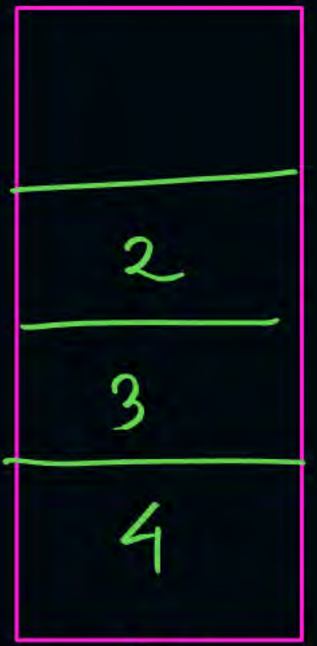
① ② ③, ④



⑤

⑥





⑤

⑦

4 is output

① Enque(1) ✓

② Enque(2)

③ Enque(3)

④ Enque(4)

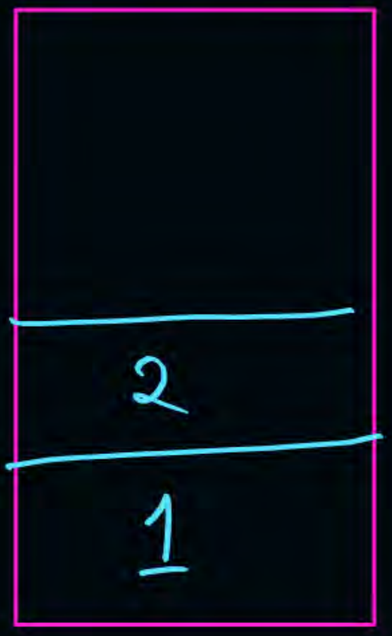
⑤ dequeue ( )

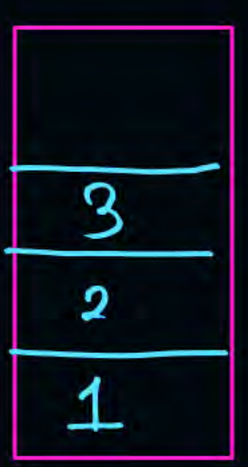
⑥ Enqueue(5)

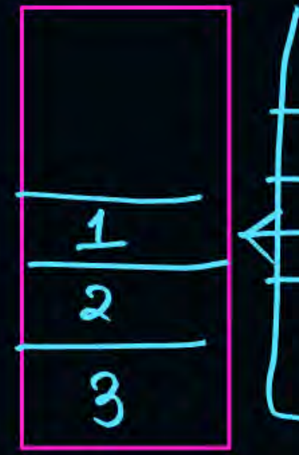
⑦ dequeue ( ) ← ②

Enqueue = Reverse, push, reverse

dequeue = pop







Q. Let Q denote a queue containing 3 numbers and S be an empty stack. Head (Q) returns the element at the head of the queue Q without removing it from Q. Similarly Top (S) returns the element at the top of S without removing it from S. Consider the algorithm given below.

Head = Front

HW

```
while Q is not Empty do
    if S is Empty OR Top (S) ≤ Head (Q) then
        X := Dequeue (Q) ;
        Push (S, x) ;
    else
        X : = Pop (S) ;
        Enqueue (Q, X) ;
    end
end
```

The maximum possible number of iterations of the while loop in the algorithm is _____ **(GATE_2016_2 M)**

Linked List

Array: fixed size static = compile time

Dynamic — Runtime

Array — ordered array — Insertion
— Delete
— Search
} shift, Binary Search

Unordered array — Insertion
Delete
Search → O(n) → worst case → size

Slide

Linked List

1 comparison $n/2$ (1 comp)

2 comparison $n/2^2$

3rd comparison $n/2^3$

⋮

kth comparison $n/2^k$

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

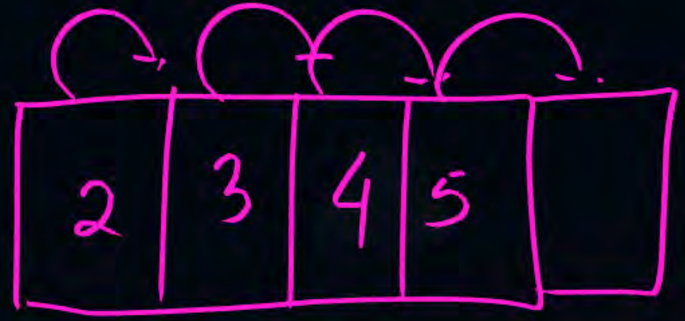
$$\log_2 n = \log_2 2^k$$

$$k \log_2 2 = \log_2 n$$

$$k = \log_2 n$$

Slide

ordered Array : insertion Deletion cause shifting of element



Insert

Linked List

Array : Compile time allocation

: Insertion Deletion (costlier)

Linked List : Run time memory allocation

Insertion Deletion required

No shift operation

Slide

Linked List

Structure is collection of dissimilar element


```
typedef struct node{
        int data;
        struct node* next;
} Node;
```

Struct node a; ← 1 way

Node a; ← alias

↑ 2nd way

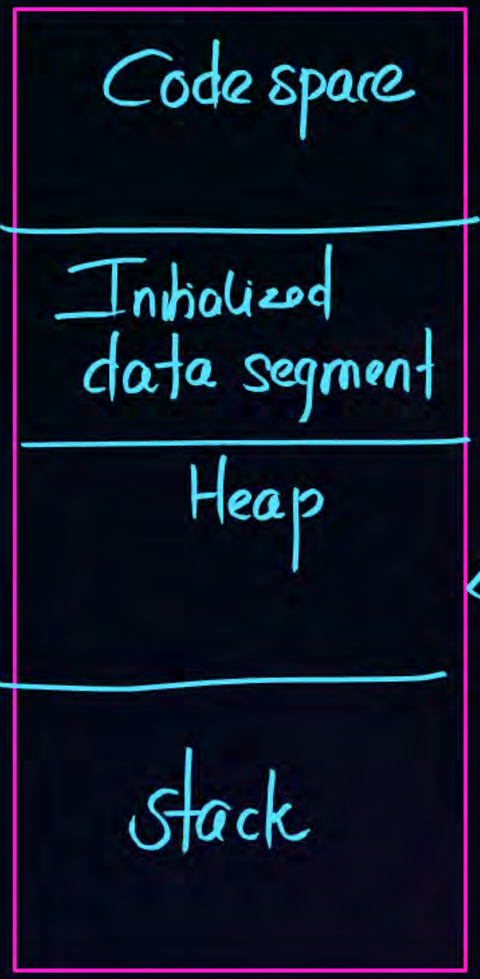
Slide

Linked List

Compile time { struct node a; ← Is this compile time/
Node a; ← Run time

Not run time

Linked List



Dynamic allocation of memory

#include<stdlib.h>

malloc() , calloc()

```
int * ptr;
ptr = (int*) malloc ( sizeof(int) );
```

Slide

Linked List

* malloc allocate 4 Bytes of space in memory & return address
* malloc return **void pointer**
* Type casting of void pointer is Not mandatory

Slide

2 mins Summary

Topic

Topic

Topic

Topic

Topic

Queue Implementation of stack

Linked List Node

.

Slide

THANK - YOU

Slide