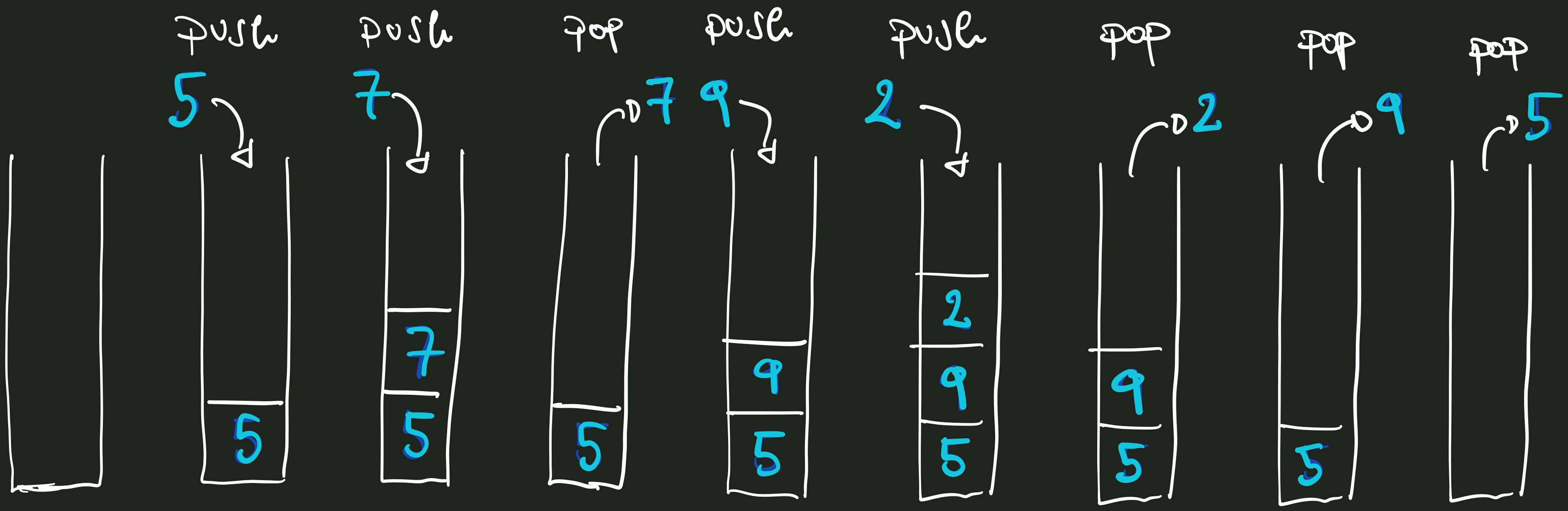
Data structures, Algorithms & Databases

Stacks, Queues and Heaps

Mirco Giacobbe

Stacks

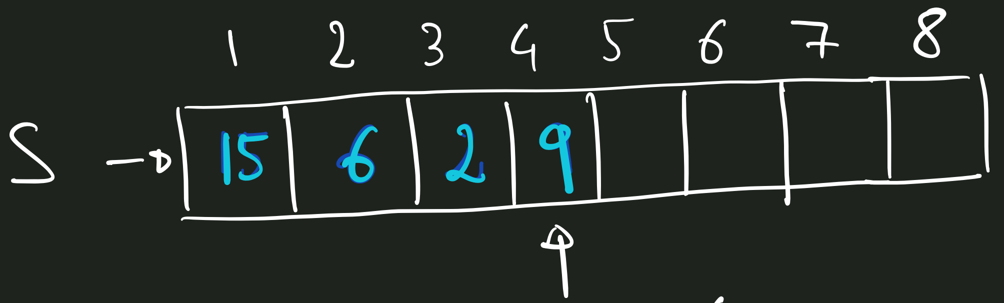


LIFO : last in, first out

Stack Operations

- Stack-Empty(S) : returns true when empty
- Push(S, x) : insert element
- Pop(S) : remove and return last element

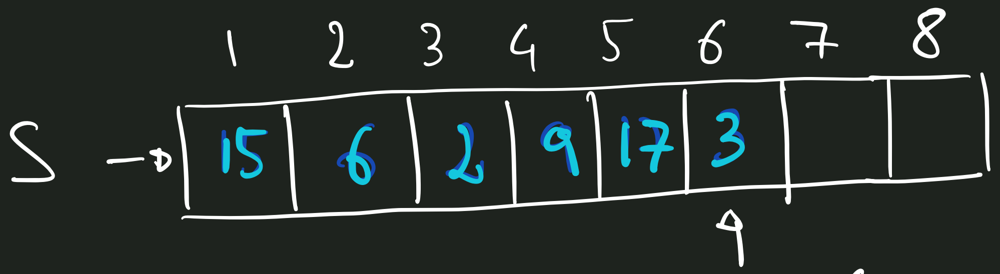
Stack Implementation Using Array



	1	2	3	4	5	6	7	8
S →	15	6	2	9				

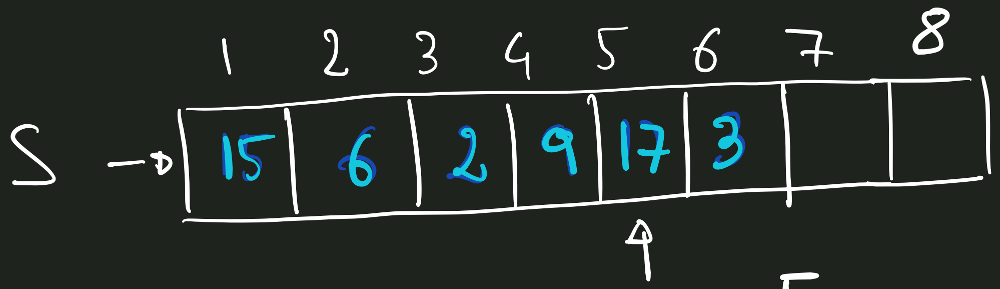
S.size = 8

S.top = 4



	1	2	3	4	5	6	7	8
S →	15	6	2	9	17	3		

S.top = 6



	1	2	3	4	5	6	7	8
S →	15	6	2	9	17	3		

S.top = 5

```
Push (S, x)
  if S.top == S.size
      error "overflow"
  else
      S.top = S.top + 1
      S[S.top] = x
```

Complexity ?

$O(1)$

S →

1	2	3	4	5	6	7	8
15	6	2	9				

↑ S.top = 4

Push (S, 17)

S →

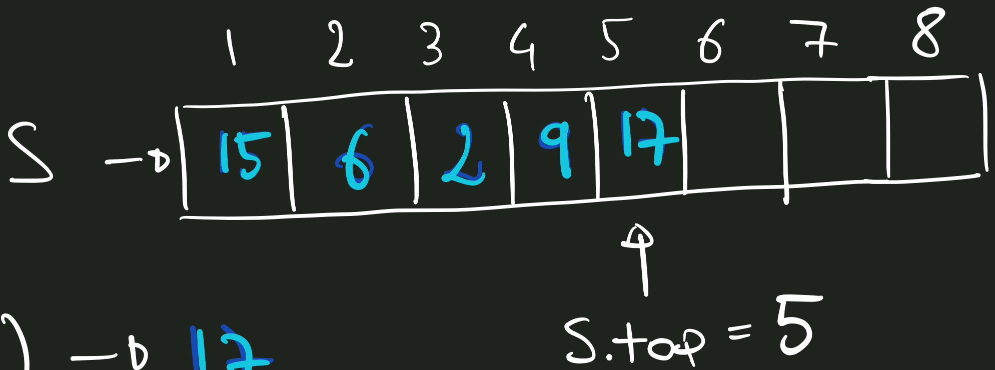
1	2	3	4	5	6	7	8
15	6	2	9	17			

↑ S.top = 5

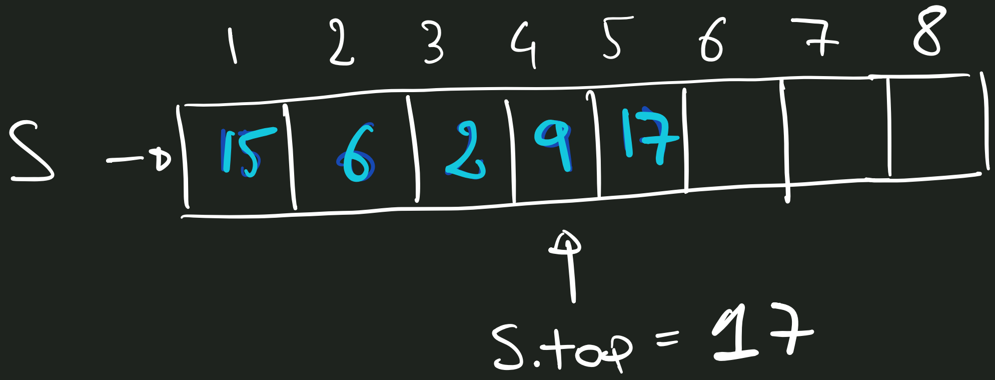
```
Pop(S):
    if S.top == 0
        error "underflow"
    else
        S.top = S.top - 1
        return S[S.top + 1]
```

Complexity ?

$O(1)$

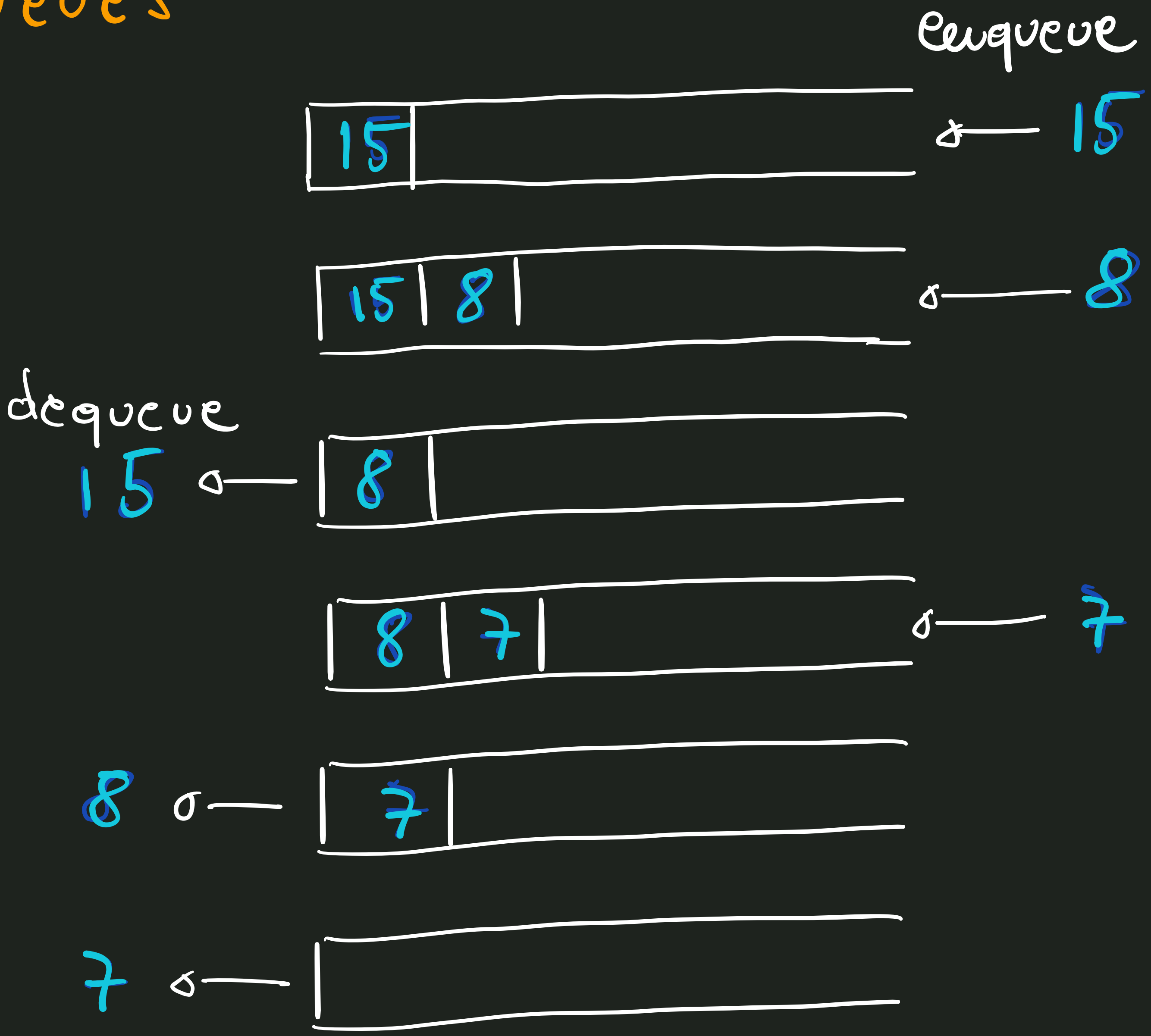


Pop(S) → 17



push	pop	stack-empty
$O(1)$	$O(1)$	$O(1)$

Queues



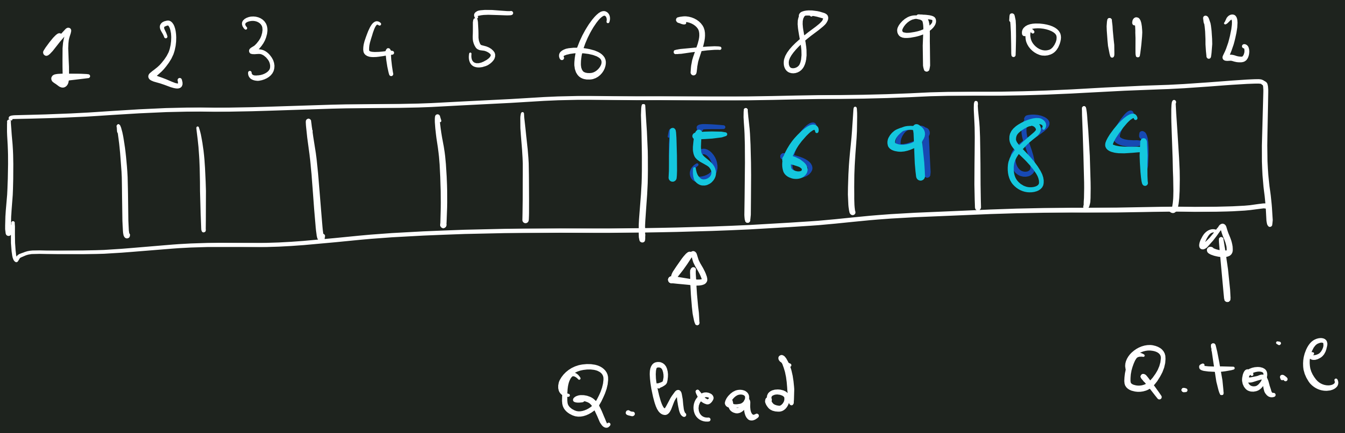
FIFO: first in, first out

Queues Operations

- Enqueue(Q, x) : insert x at the tail of the queue Q

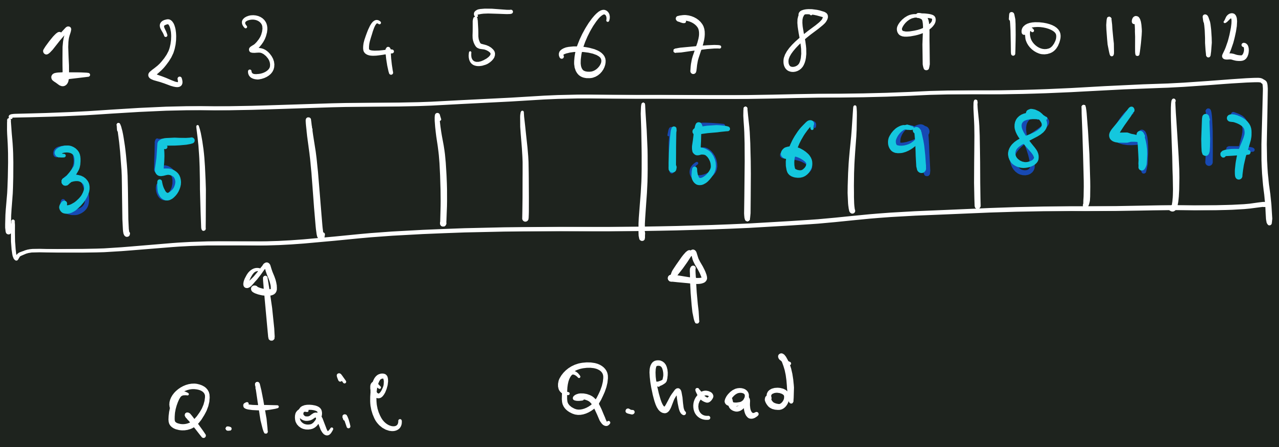
- Dequeue(Q) : remove and return the head of the queue Q

Implementation using Array:



1	2	3	4	5	6	7	8	9	10	11	12
						15	6	9	8	4	

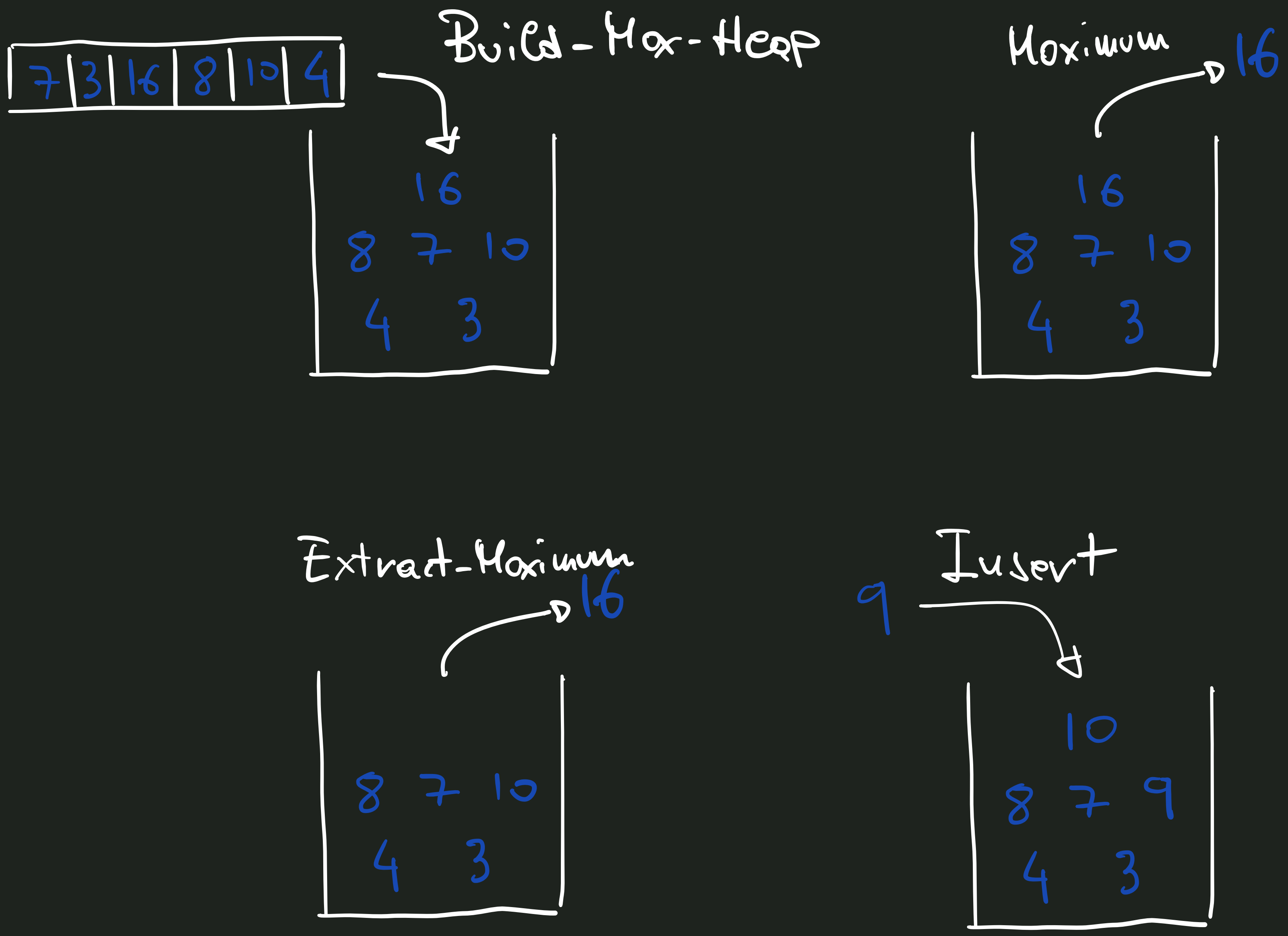
Q.size = 12



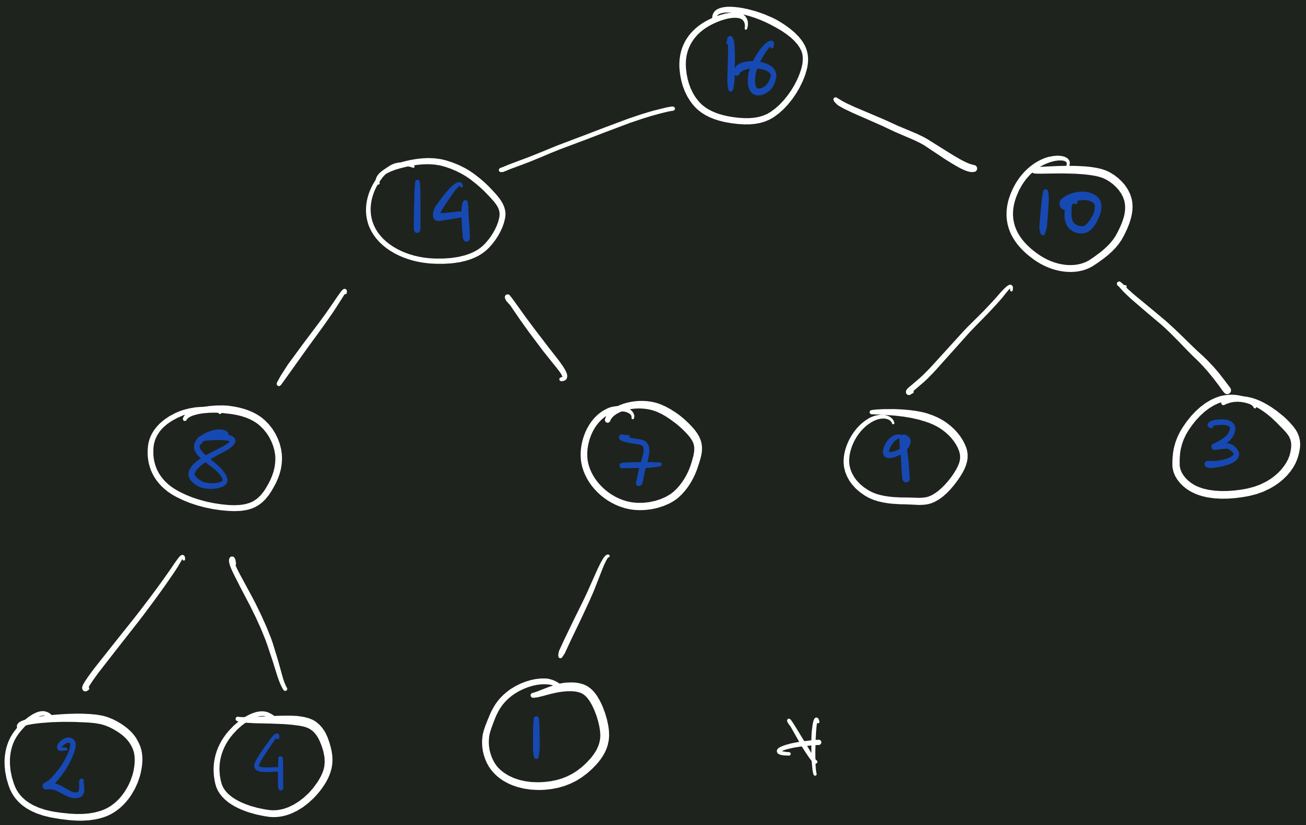
1	2	3	4	5	6	7	8	9	10	11	12
3	5					15	6	9	8	4	17

Enqueue	Dequeue
O(1)	G(1)

Max-Heaps



7 | 3 | 16 | 8 | 10 | 4
Build-Max-Heap
16
8 7 10
4 3
Maximum
16
16
8 7 10
4 3
Extract-Maximum
16
8 7 10
4 3
Insert
9
10
8 7 9
4 3



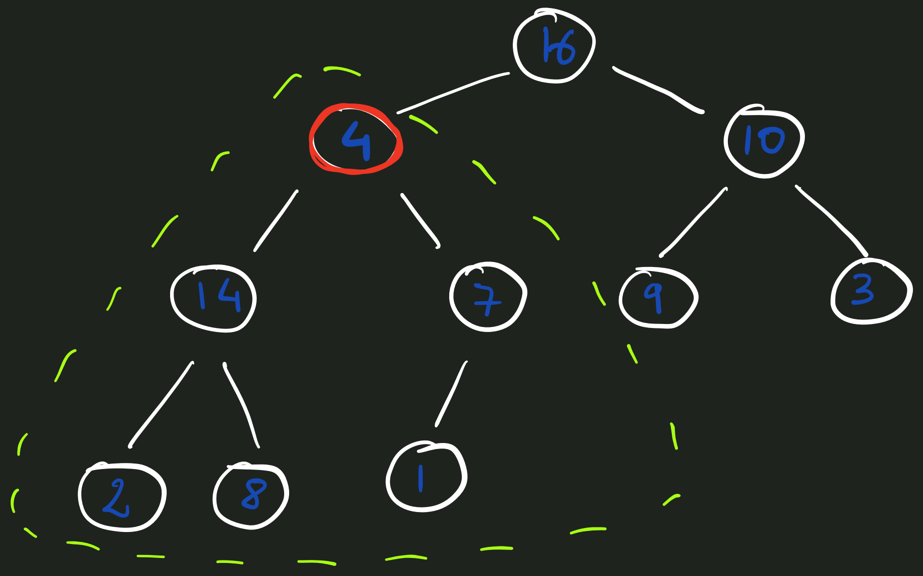
Max-Heap Property:

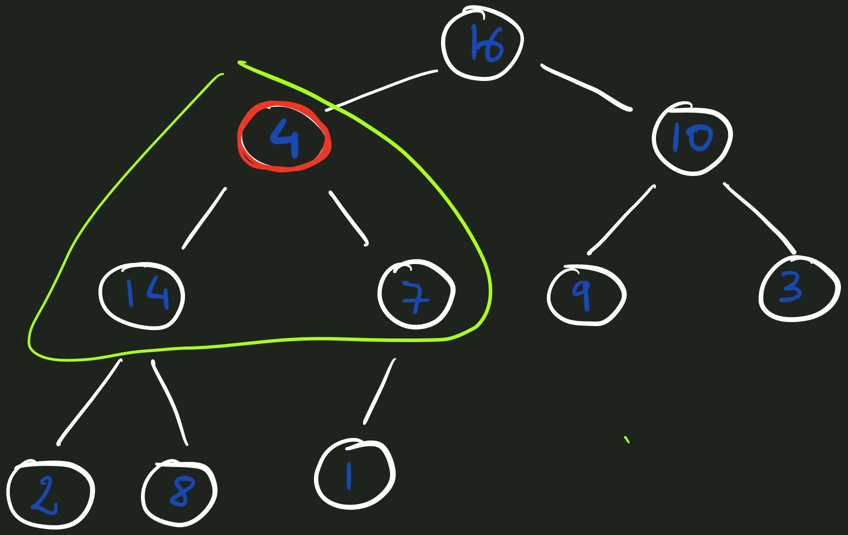
value of parent ≥ value of child

Mox-Heapify Procedure

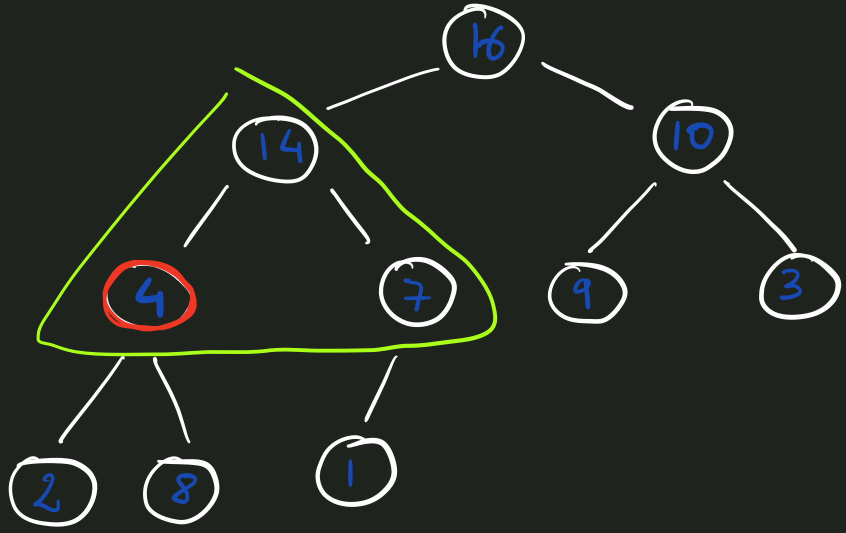
- Enforces the max-heap property of a node N

- Assumes that the children of N are valid max-heaps

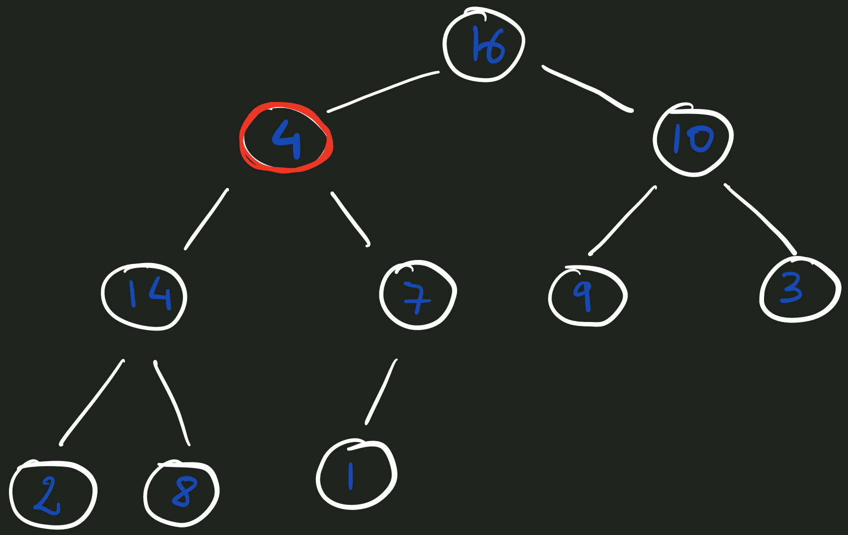

16
4
10
14
7
9
3
2
8
1

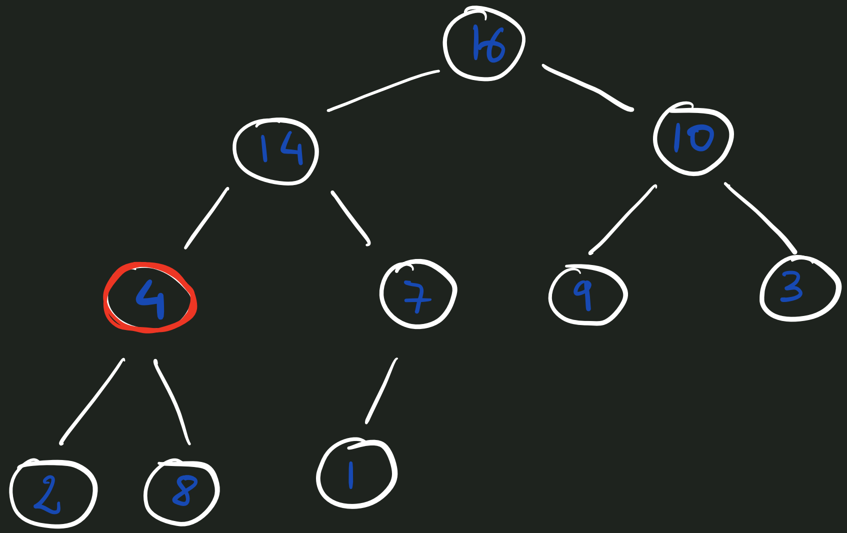


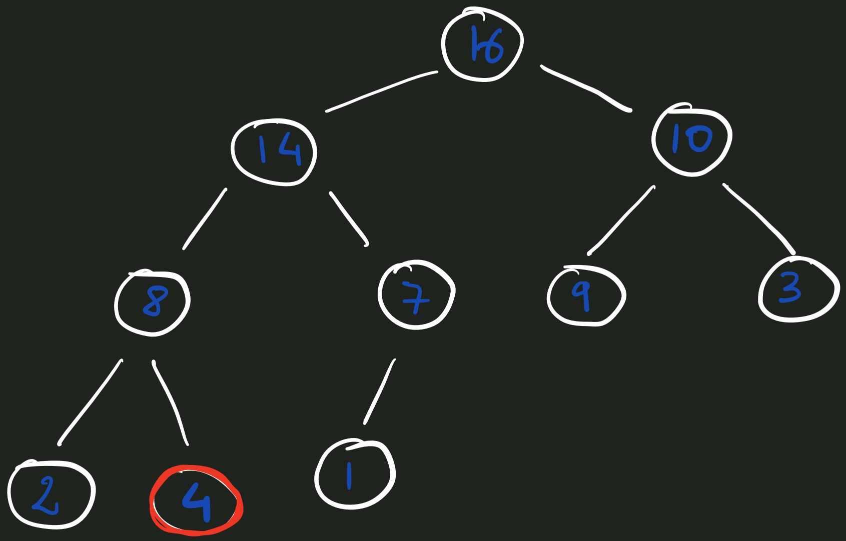
pick the maximum



swap


16
4
10
14
7
9
3
2
8
1


16
14
10
4
7
9
3
2
8
1


16
14
10
8
7
9
3
2
4
1

Time Complexity of Max-Heapify

Let h be the height of the tree

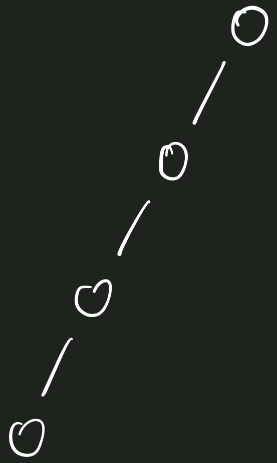
- How many swaps in the worst case, as a function of h?

$$O(h)$$

How many nodes does a binary tree of height h contain?

How many nodes does a binary tree of height h contain?

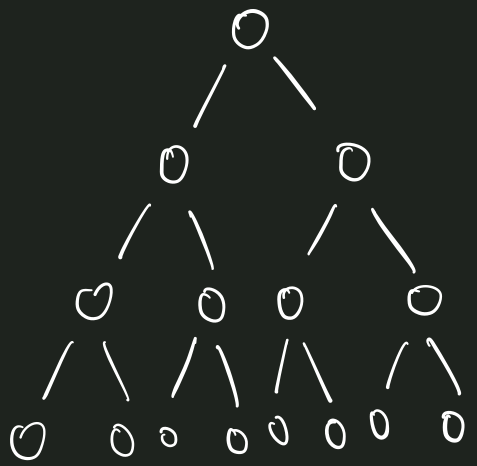
h	$\#n$
1	1
2	2
3	3
4	4



at least h nodes

How many nodes does a binary tree of height h contain?

h	$\# n$
1	1
2	3
3	7
4	15



at most $2^{h} - 1$ nodes

number of nodes

$$h \leq n \leq 2^{h}$$

height

when completely skewed

when perfectly balanced

Logarithm in base 2

Definition : $\log_2 n$ is the number k such that $2^k = n$

n	1	2	4	8	16	…	131072	262144
$\log_2 n$	0	1	2	3	4		17	18

$\log 250000$ → somewhere inbetween

height

=

worse case time complexity of Max-Heapify

$$\log n \leq h \leq n$$

when perfectly balanced

when completely skewed

Max-Hepify
$O(\log n)$