Stack push pop

10 20 30
30 20 10

peek

top → 40 30 20 10

S.pop() → 50
S.push(40)
S.peek() → 30

isEmpty() isFull()

Stack

top = -1 ← 0 1 2 3 4

size = 4
top → 0 1 2 3 4

isEmpty
top = -1

isFull
top == size - 1

push(data) — enq pop — deq, front, end

Queue

Stack

Queue

push(data)
insert at pos 1
return data

pop
del at pos 1
return data

Dual Stack

Size 10

Size

top = 20
top1 = 18
top2 = 19

top1 = -1
top2 = size

isEmpty
top1 == -1 && top2 == size

isFull
top1 + 1 == top2

Dual Stack
int t[] arr;
size
top1, top2

push(data)
if isFull
return false
arr[top1] = data
top1--
return true

push(data)
if isFull()
return false
arr[top2] = data
top2++
return true

Queue

front rear

a b c d
— insert isEmpty
— delete isFull

Queue
int front
int rear
int
Size = 5

isEmpty
f == -1 || r == -1
||
f > r

isFull
r == size - 1

f(1)

Circular Queue

Back Front Solving

R

F

F

R

F

R

F

R

F

R

F

R

isEmpty
f == -1 || r == -1
F == R

isFull
f == -1 || R == R-1
||
(R+1) % S == F

last op is delete
F = R
isEmpty

last op is insert
F = R
isFull

F = -1

last op = 'I'

Tree

root

n1 n1-1
children edges

Size

Level

Level of tree — 3

Binary Tree

2^{l-1} (l)

2^{l-1} (l)

2^{l-1} (l)

$(2^{h+1}) - 1 = 7$

Tree Traversal

Preorder a b c
Inorder b a c
Postorder b c a

Pre 50 10 20 40 15 60 80 85 70 30
In 20 10 15 40 50 80 70 60 30
Post 20 15 40 10 85 70 80 60 30 50

BFS

10 20 30 40 50 60 70

10 20 30 40 50 60 70