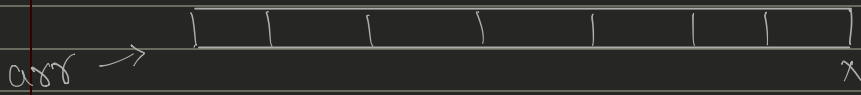
k - Queues in an Array ->

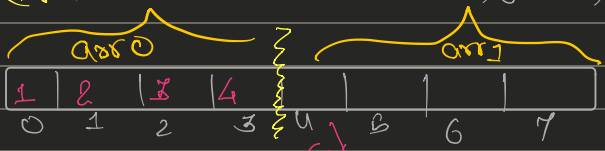


N = 12
k = 3

k - Queues

(M-1) -> Brute Force ->

N = 8, k = 2

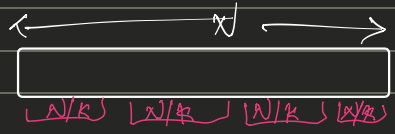


push(5, Q0) Can't fill here

=> N/k = 8/2 = 4

arr_0 -> Q_0

arr_1 -> Q_1



k - Queues -> to be imple.
=> (N/k) parts of arr.

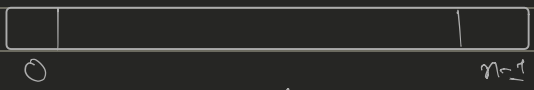
Fragmentation Problem

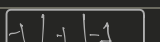
(Discarded Method)

=> Not Space Optimal

M-2 -> Efficient Method in terms of space ->

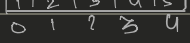
(1) -> arr



front [k] ->  -> front index of $q_{i^{th}}$

rear [k] ->  -> rear index of $q_{i^{th}}$

next [n] -> stores indexes of next item for all items in arr



free spot -> (initialize with zero)
↳ current available free spot

Push(x, i^{th}) ->

(1) -> overflow -> if (freespot == -1){
return false
}

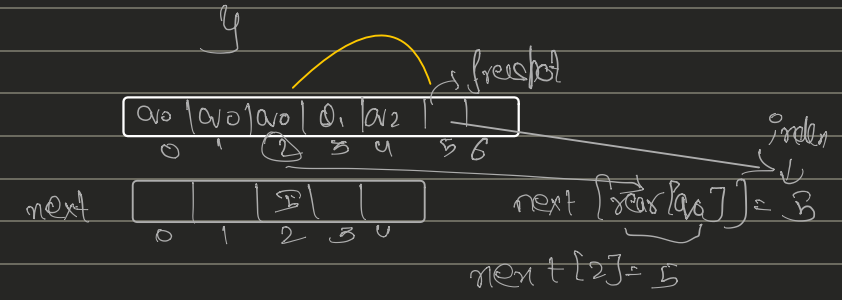
(2) -> // find first free index
int index = freespot;

(3) -> // update free spot -> next[index]

(4) -> if first element $q_{i^{th}}$
if (front[q.n] == -1){
front[q.n] = index;
}
else
// link new element to rear

Queue nearest element

next [rear [qn]] = index



// update next

next [index] = -1

// update rear

rear [qn] = index

arr [index] = x;

return true

(1) => POP [qwith] =>

(1) => // Underflow

if (front [qwith] == -1) {

return -1

}

(2) => // find index

int index = front [qn];

(3) => // front ko update kro

front [qn] = next [index]

(4) => manage free spot

next[index] = freespot
freespot = index

Return arr[index]

DRV — RUN -> N = 8, k = 3

arr

1	2	3					
0	1	2	3	4	5	6	7

next

1	2	3	4	5	6	7	-1
0	1	2	3	4	5	6	7

front

-1	-1	-1
a_0	a_1	a_2

rear

-1	-1	-1

freespot = 0 1 2 3 4 5 6 7 0

(1) -> Push(1, 0)

(i) freespot ≠ -1 -> ✓

(ii) -> index = freespot = 0
freespot = next[0] = 1

(iii) -> front[0] = -1 -> For 1st element

front[0] = 0

(iv) -> next[0] = -1

(v) -> rear[0] = 0

(vi) -> arr[0] = 1

(2) -> Push (2,0)

(i) -> No overflow

(ii) -> index = 1 , freespot = next[1] = 2

(iii) -> front[0] != -1 => so no first element

next[rear[0]] = index

next[0] = 1

(iv) -> next[1] = -1

(v) -> rear[0] = 1

(vi) -> arr[1] = 2

(3) -> Push (5,1)

(i) -> Overflow -> No

(ii) -> index = freespot = 2

freespot = next[index] = 5

(iii) -> first element -> ✓

front[1] == -1 -> ✓

So, front[1] = index = 2

(iv) -> next[index] = -1

next[2] = -1

(v) -> rear[1] = index = 2

(vi) -> arr[2] = 5

(4) -> Push (8,0)

(i) -> No Overflow

(ii) -> index = freespot = 3
freespot = next[3] = 4

(iii) -> first element -> X

next [rear[0]] = index
next [1] = 3

(iv) -> next [index] = next [3] = -1

(v) -> rear [qi] = index
rear [0] = 3

(vi) -> arr [3] = 8

(5) -> Pop (1) ->

(i) -> underflow -> X
front [qi] = front [1] = -1 X

(ii) -> index = front [qi] = 2

front [1] = next [index] = next [2] = -1

(iii) -> freespots
next [2] = 4
freespot = 2

(iv) -> Return arr [2] -> 5

(6) -> Pop(0) ->

(i) -> front[0] != -1 no underflow

(ii) -> index = front[0] = 0

front[0] = next[0] = 1

(iii) -> freespot

next[index] = freespot = 2

freespot = index = 0

(iv) -> Return arr[0] -> 1