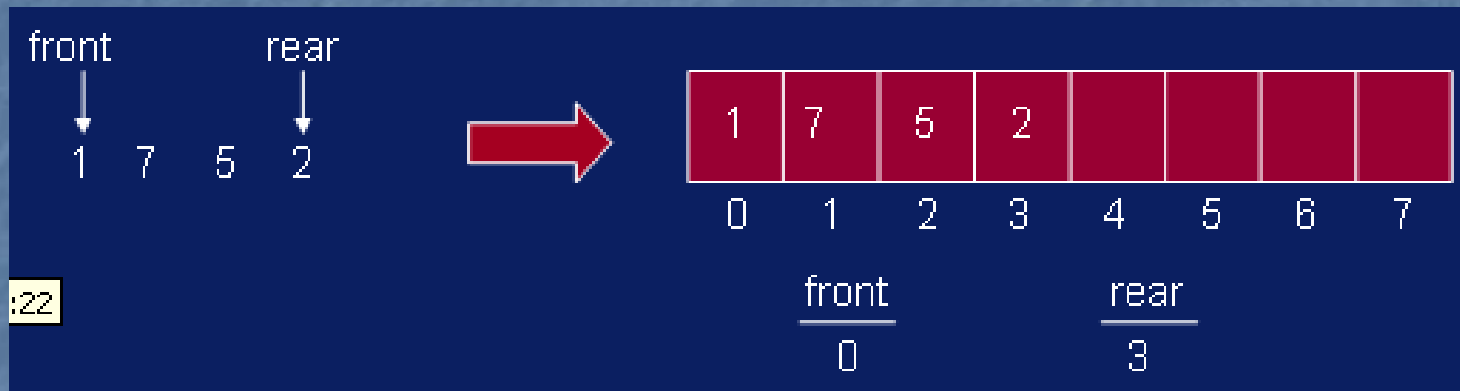


# Lecture # 8

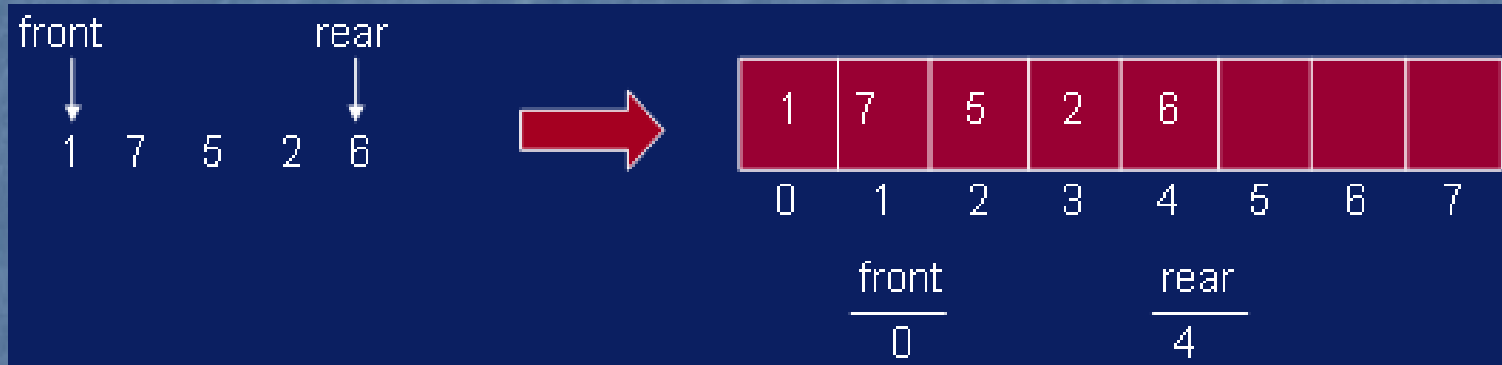
# Queue using Array

- If we use an array to hold queue elements, both insertions and removal at the front (start) of the array are expensive.
- This is because we may have to shift up to “n” elements.
- For the stack, we needed only one end; for queue we need both.
- To get around this, we will not shift upon removal of an element.

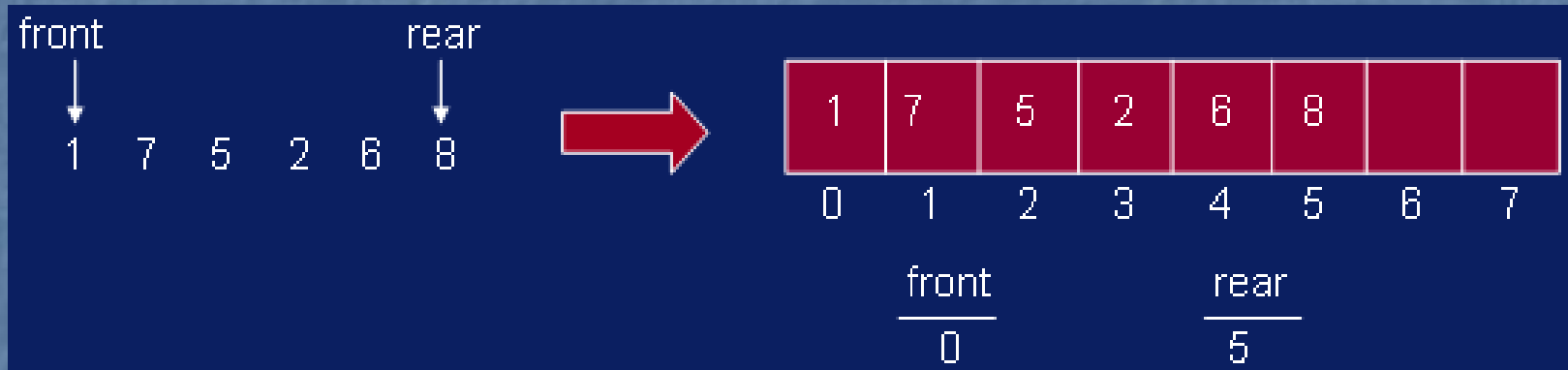
# Queue using Array



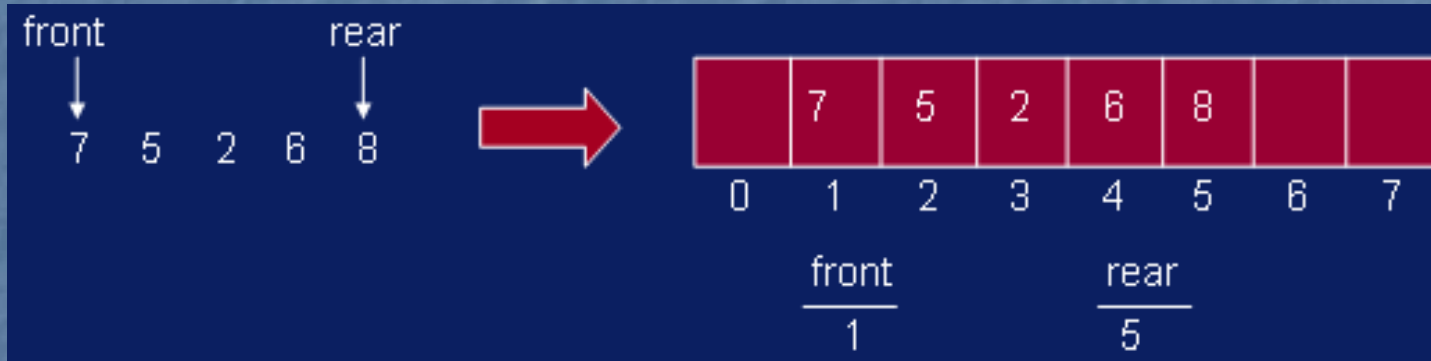
# Queue using Array



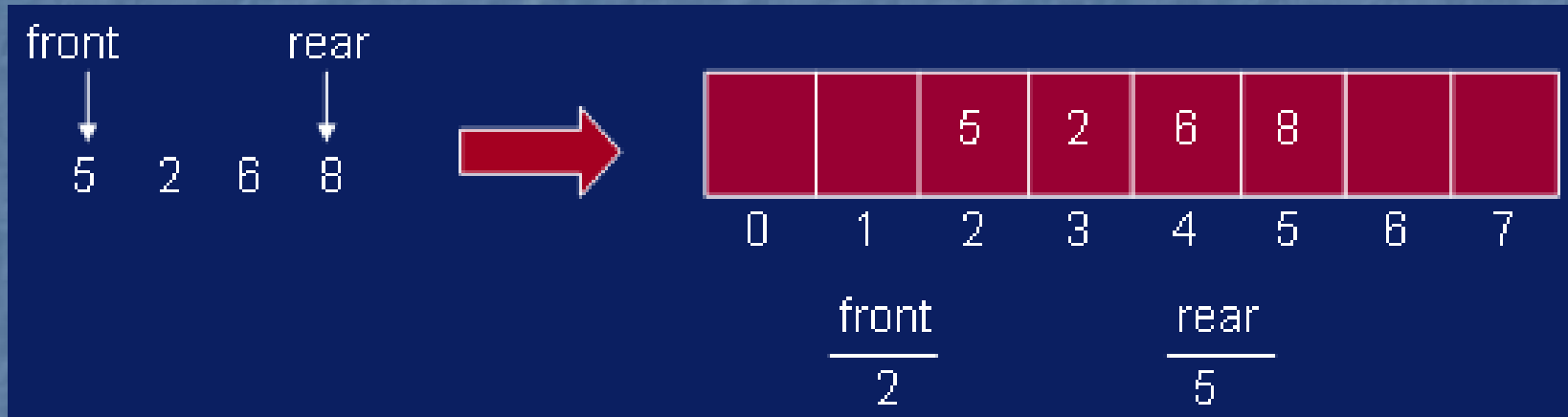
# Queue using Array



# Queue using Array



# Queue using Array





# Queue using Array

enqueue(9)  
enqueue(12)

front  
↓  
5   2   6   8   9   12  
rear  
↓



		5	2	6	8	9	12
0	1	2	3	4	5	6	7
front		rear					
2		7					

enqueue(21) ??

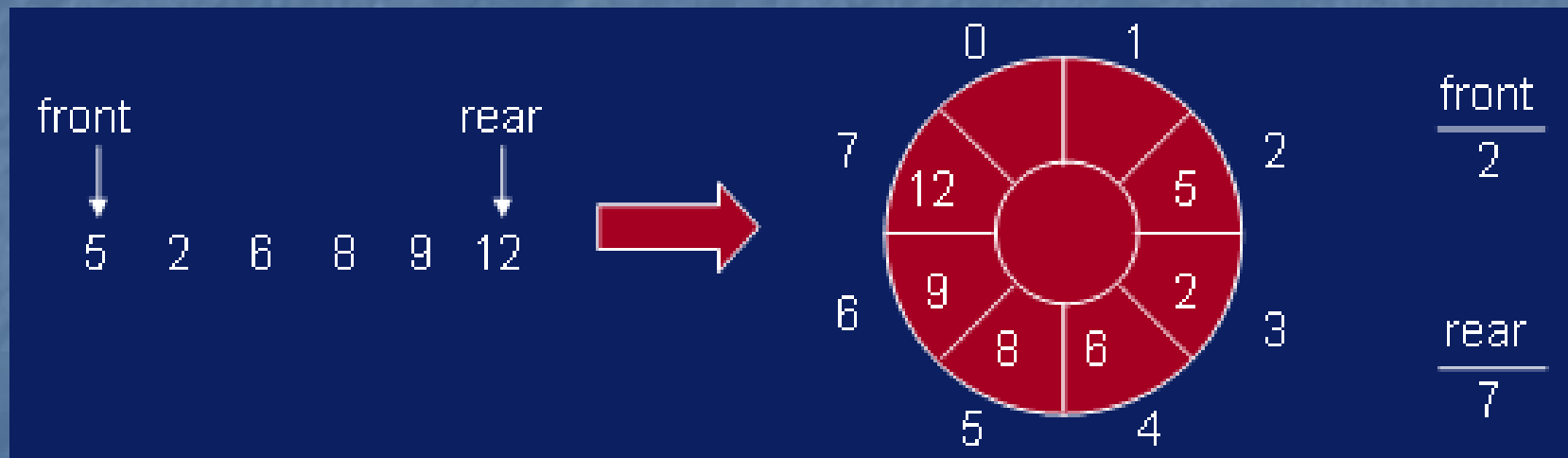


# Queue using Array

- We have inserts and removal running in constant time but we created a new problem.
- Cannot insert new elements even though there are two places available at the start of the array.
- **Solution**: allow the queue to “wrap around”.

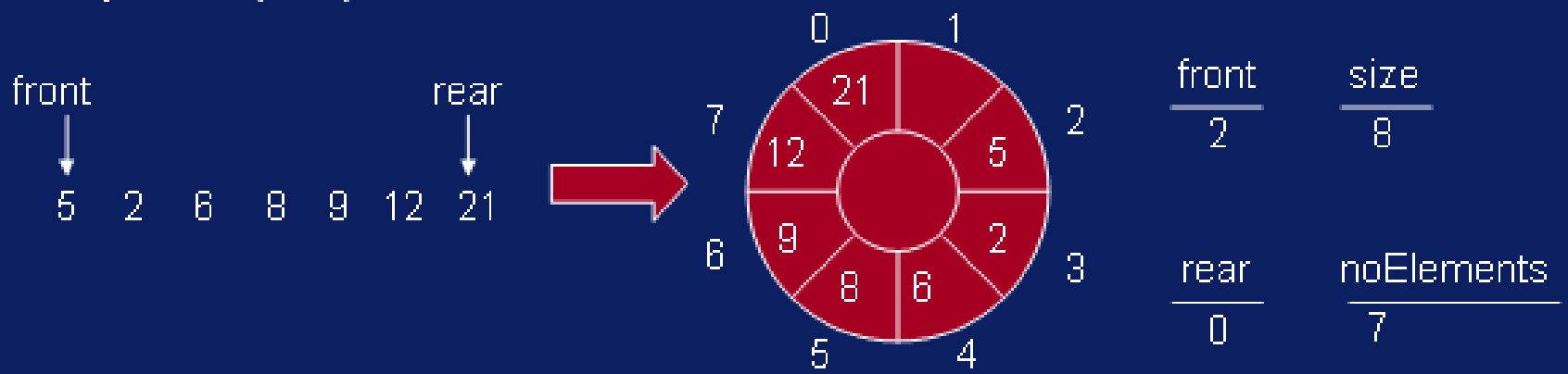
# Queue using Array

- Basic idea is to picture the array as a *circular array*.



# Queue using Array

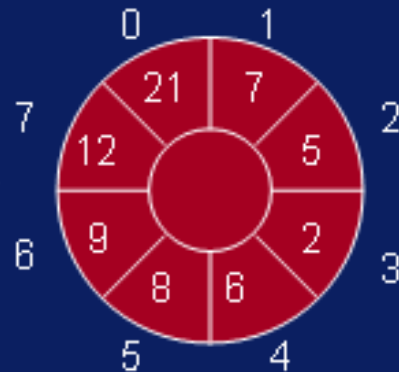
enqueue(21)



# Queue using Array

enqueue(7)

front  
↓  
5   2   6   8   9   12   21   7  
rear  
↓



front  
2

size  
8

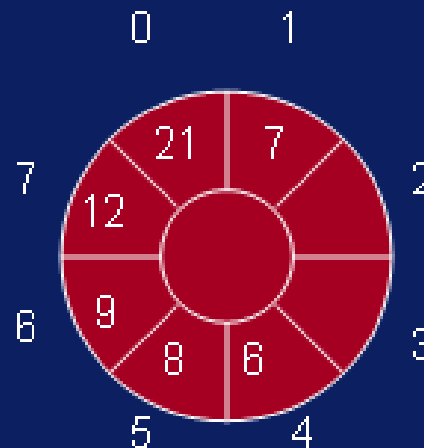
rear  
1

noElements  
8

# Queue using Array

dequeue()

front  
↓  
6   8   9   12   21   7  
rear  
↓



front  
4

size  
8

rear  
1

noElements  
6