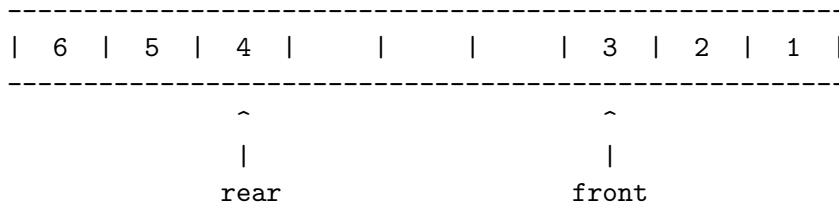


# deque

December 9, 2022

## 1 Implement a deque with an array

To implement this datatype we should use something called: **circular array**. This is not a specific array, but we store two pointer to recognise where is the front and where is the rear of our deque



### 1.0.1 Datatype specifications:

datatype: deque

methods: appendleft, append, popleft, pop

attributes: underlying\_carray

- Implementation of **appendleft**:
  - if the array is full, it is not possible
  - if the array is empty, it means front and rear are -1, so increment them and set:  
`self.underlying_carray[front] = x`
  - else, decrement front and set: `self.underlying_carray[front] = x`
- Implementation of **append**:
  - if the array is full, it is not possible
  - if the array is empty, it means front and rear are -1, so increment them and set:  
`self.underlying_carray[rear] = x`
  - else increment rear and set: `self.underlying_carray[rear] = x`
- Implementation of **popleft**:
  - if the array is empty, return
  - if there is only one element in the deque, store `to_ret = self.underlying_carray[front]` and set: `front, rear = -1, -1` and return `to_ret`
  - else store `to_ret = self.underlying_carray[front]` and increment front by one and return `to_ret`
- Implementation of **pop**:
  - if the array is empty, return
  - if there is only one element in the deque, store `to_ret = self.underlying_carray[rear]` and set: `front, rear = -1, -1` and return

```
    to_ret
- else store to_ret = self.underlying_carray[rear] and decrement rear by one and
  return to_ret
```