Circular Queue

{ Private:

queue[]

front

rear

capacity

size

};

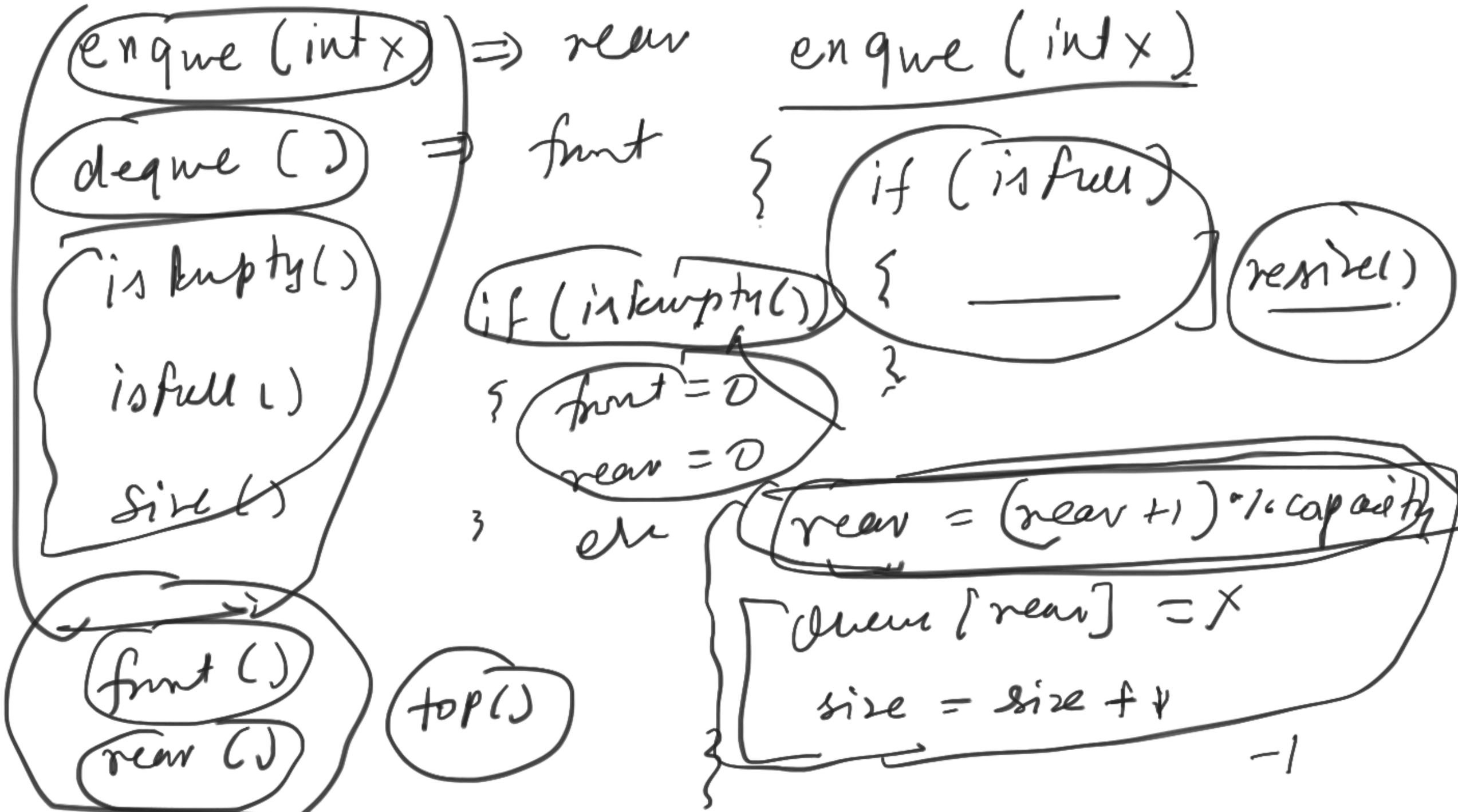
queue [maxSize]

front = -1

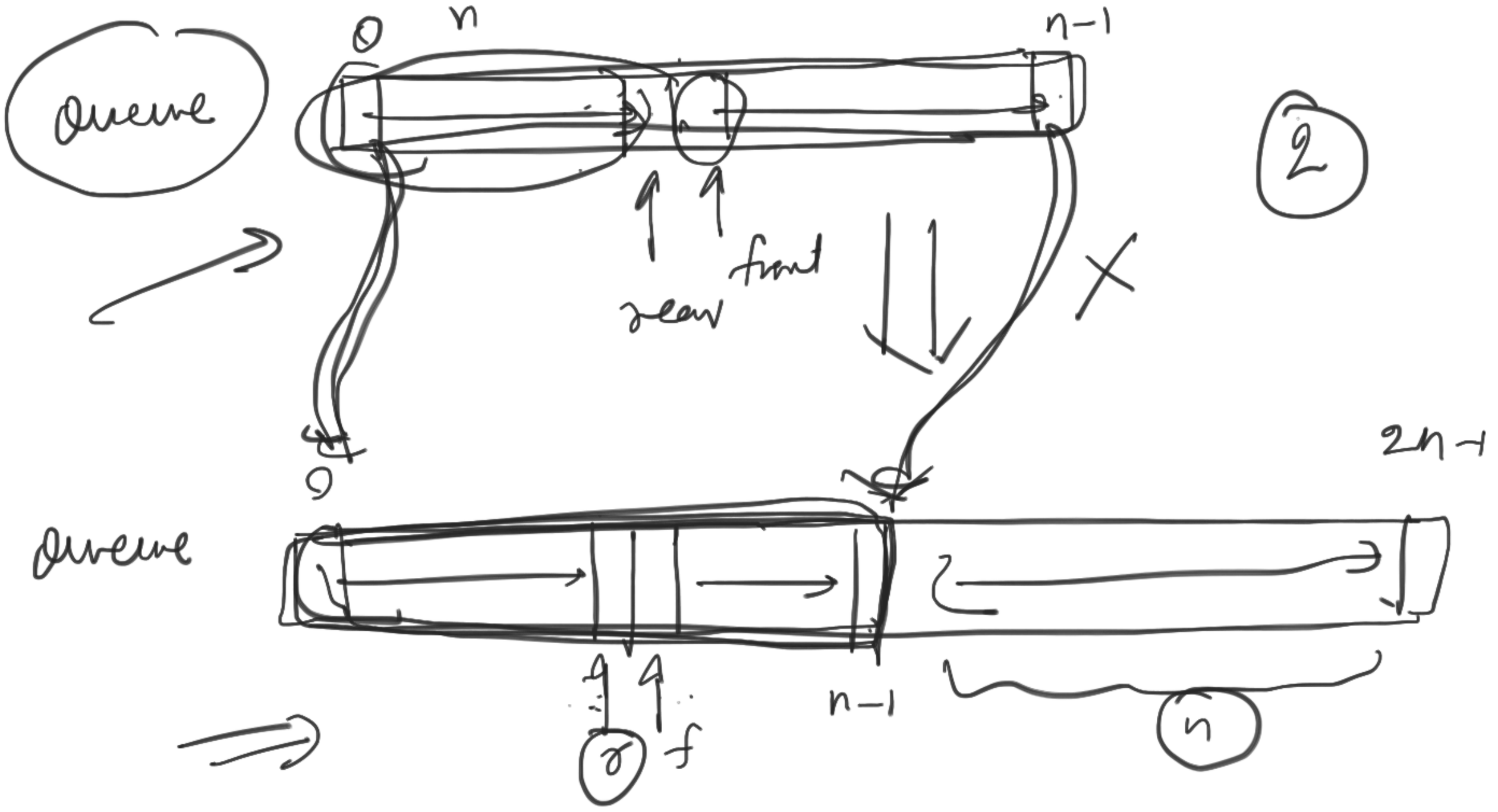
rear = -1

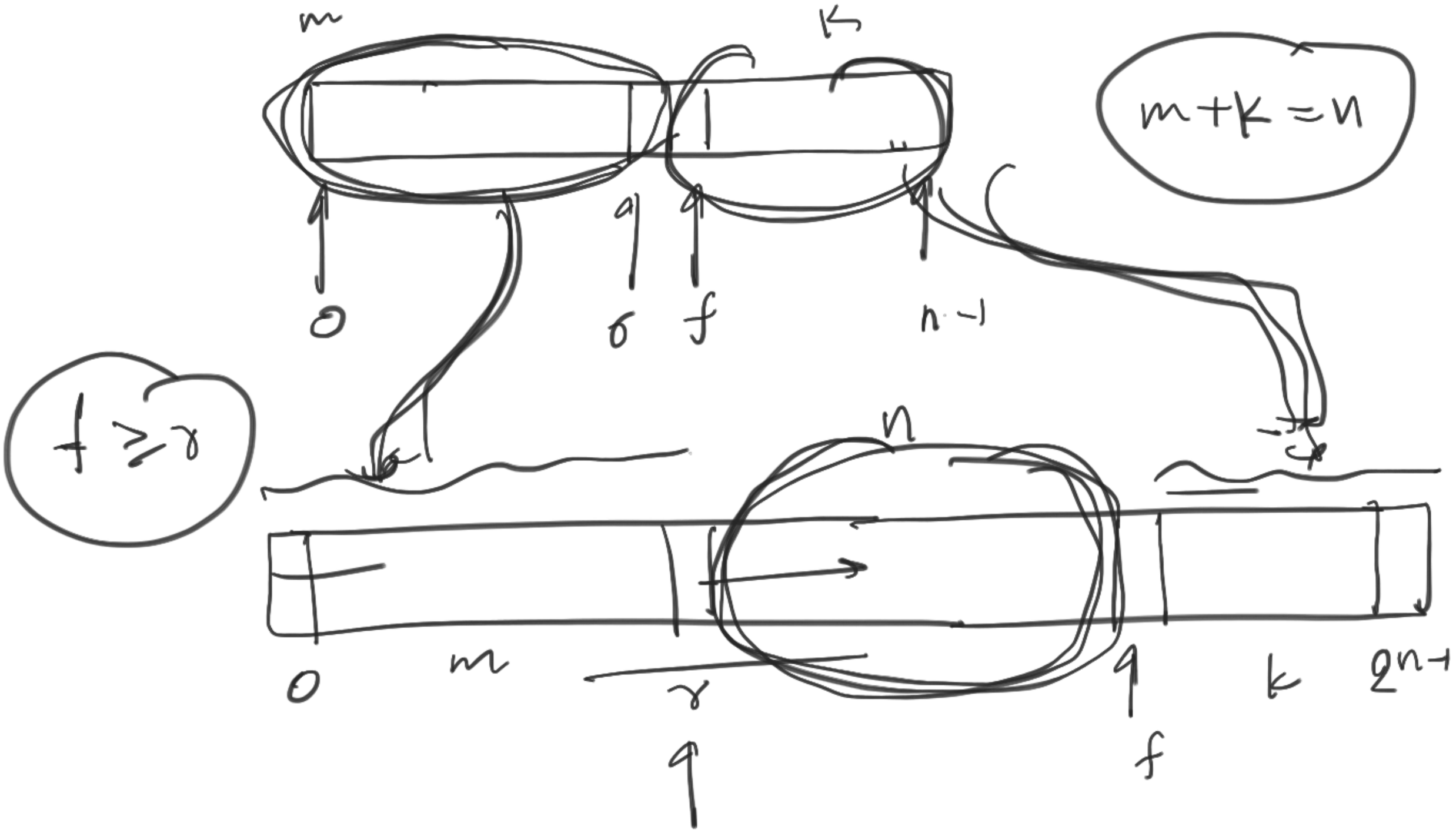
size = 0

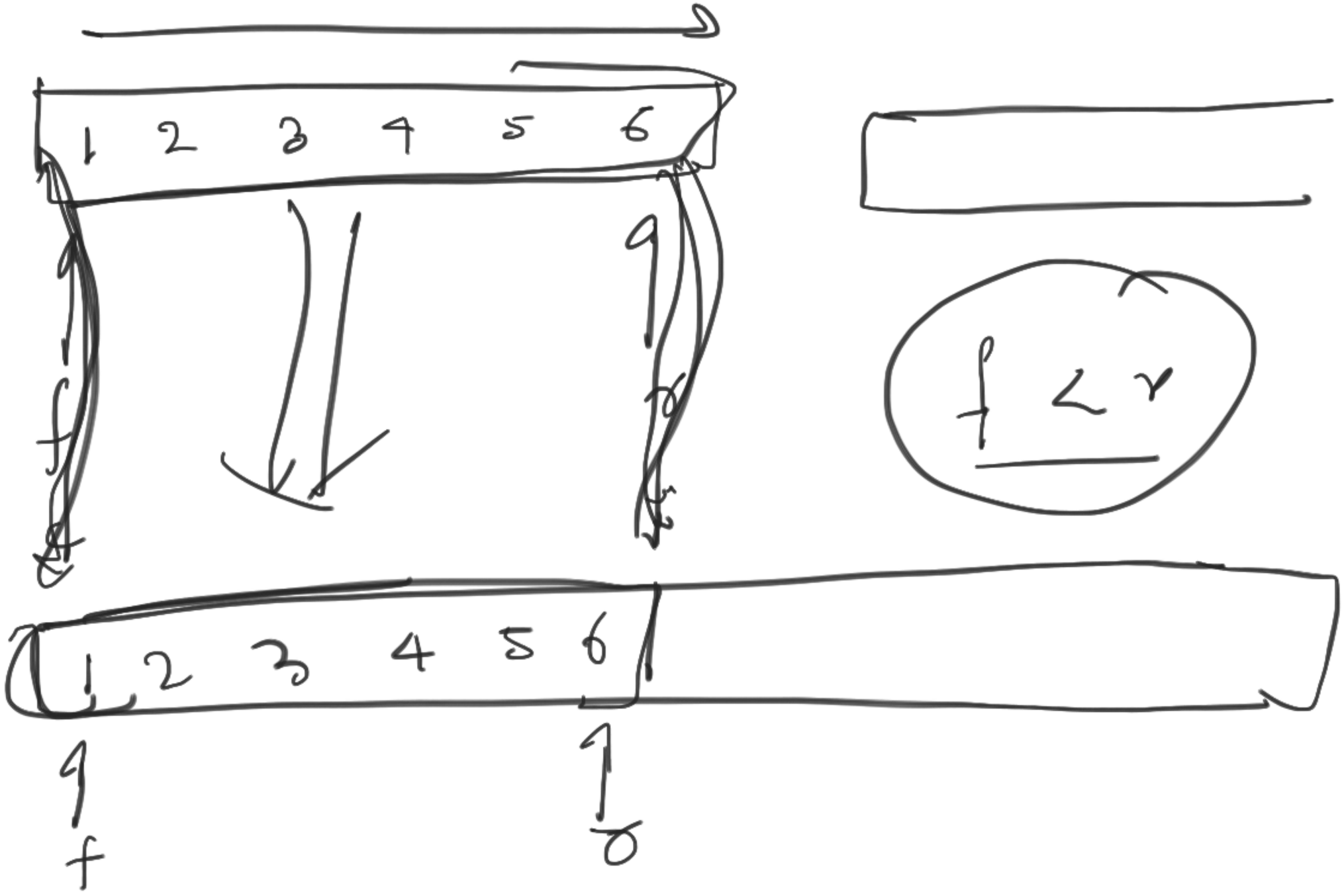
capacity = maxSize

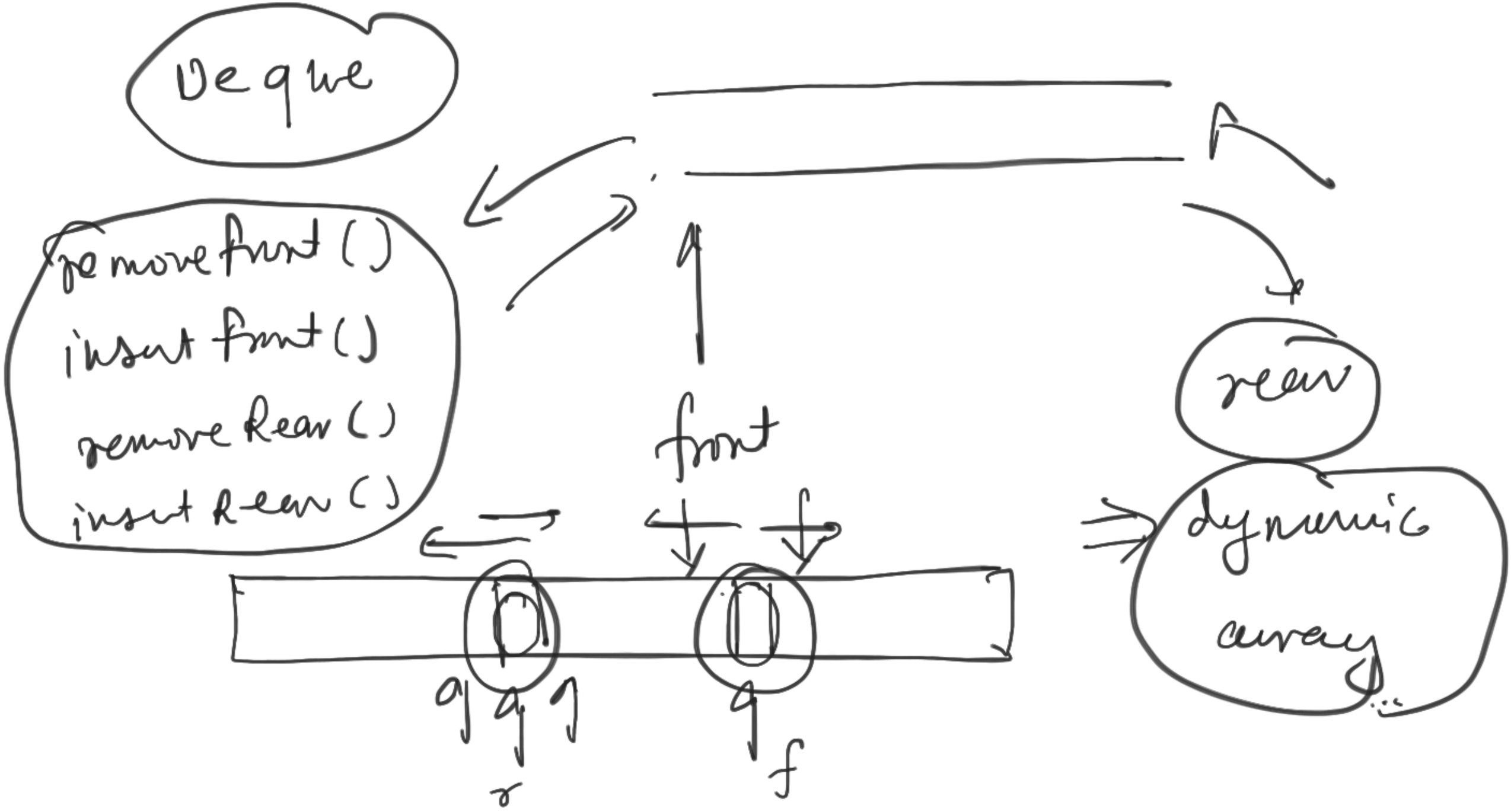

enque(int x)
dequeue()
isempty()
isfull()
size()
front()
rear()
top()
⇒ rear
⇒ front
enque(int x)
{
if(isfull)
{ ——— }
resize()
if(isempty())
{ front = 0
rear = 0 }
else
rear = (rear+1) % capacity
queue[rear] = x
size = size++
}
-1

```
int dequeue()
{
    if( isEmpty() )
    {
        ____________
    }
    temp = queue[front]
    if( front == rear )
    {
        front = -1
        rear = -1
    }
    else
        front = (front+1) mod capacity
    return temp
}
```

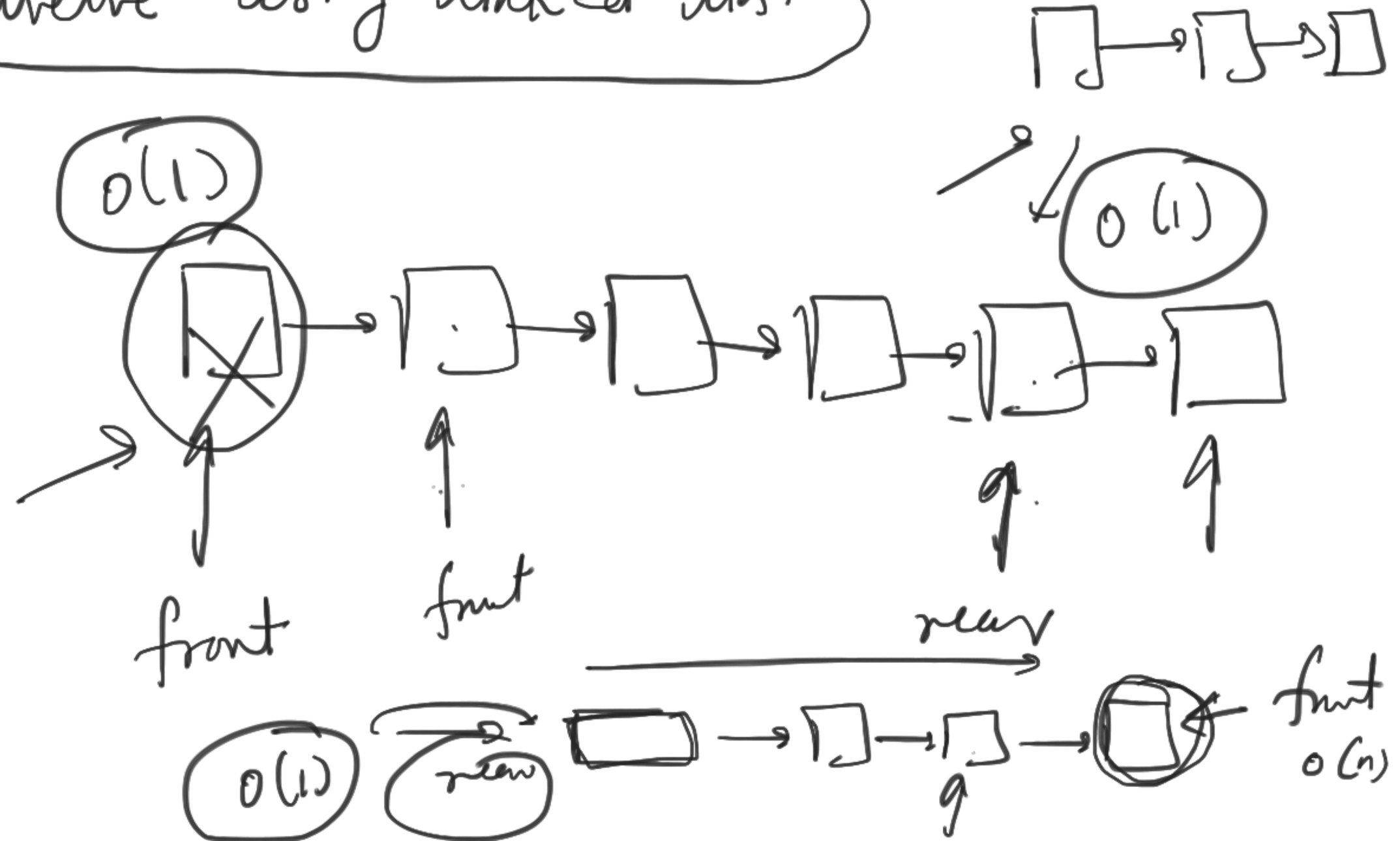

Queue
0
n
n-1
2
rear
front
X
0
2n-1
Queue
n-1
n
r
f

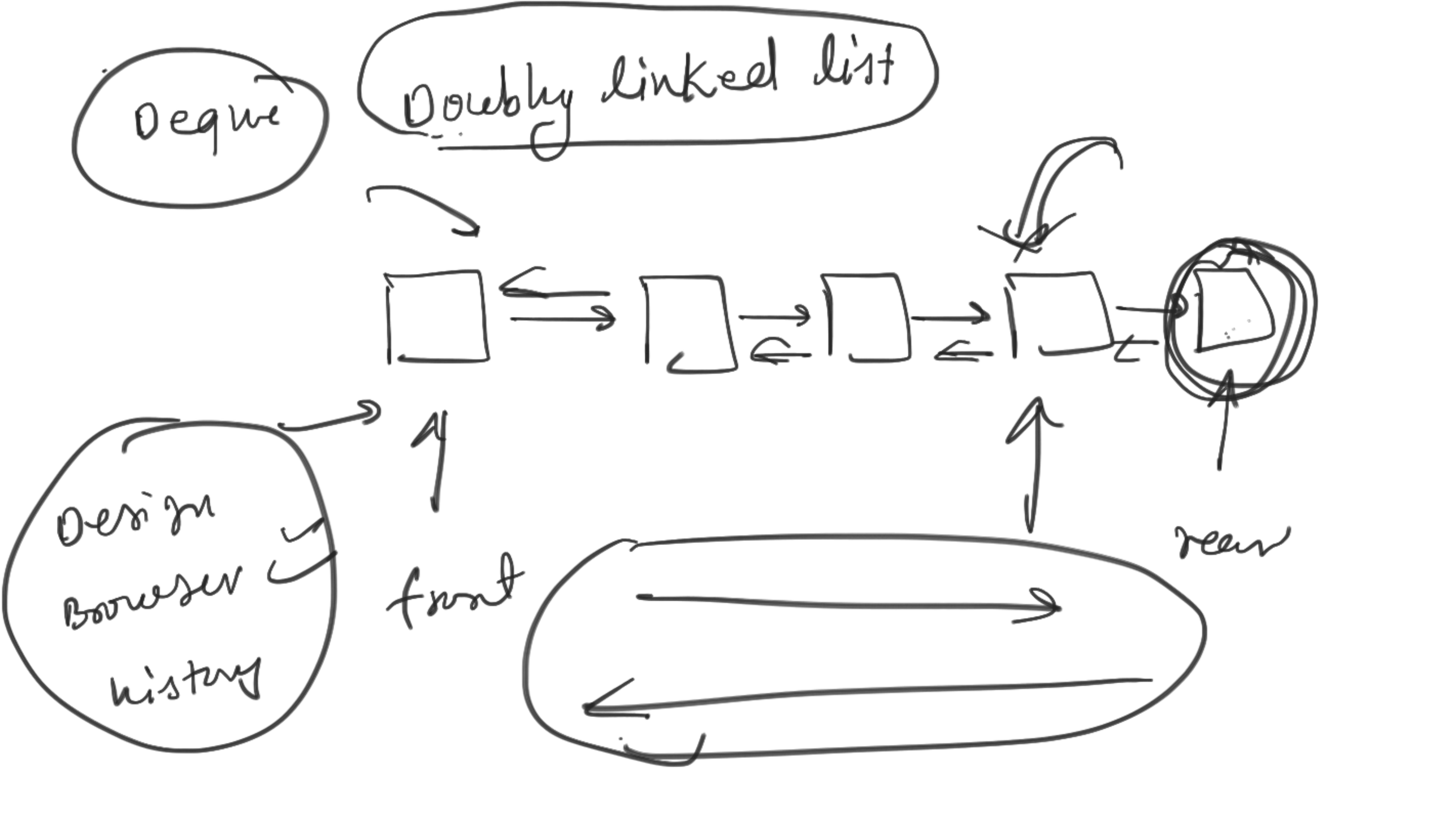

m
k
m+k=n
0
q
f
n-1
f ≥ γ
n
0
m
γ
q
f
k
2^n-1

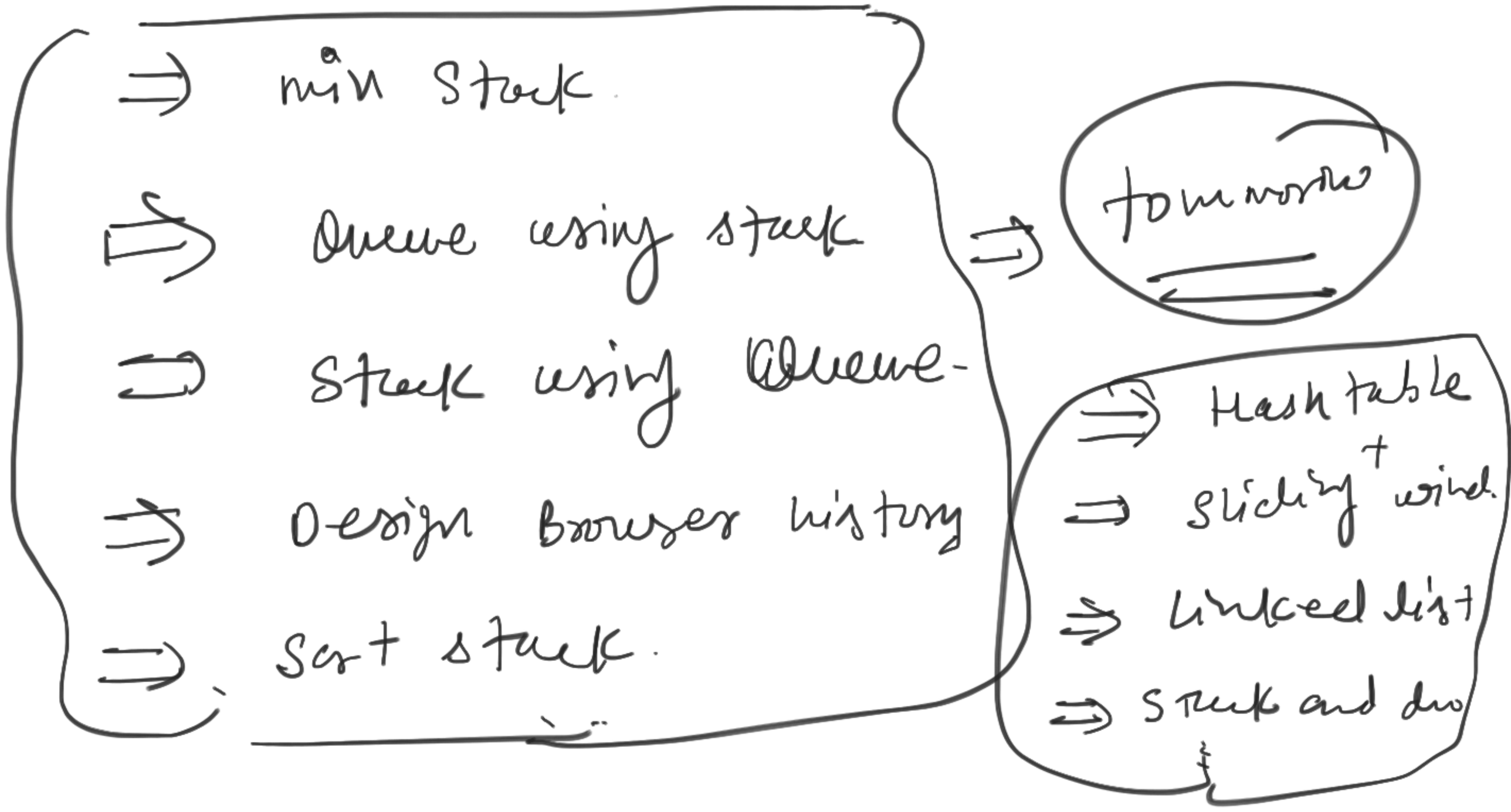

1 2 3 4 5 6
f
f < r
1 2 3 4 5 6
f
δ


Deque
removeFront()
insertFront()
removeRear()
insertRear()
front
rear
dynamic
array
q
r
f

Queue using linked list




Deque
Doubly linked list
Design Browser history
front
rear


⇒ min Stack
⇒ Queue using stack
⇒ Stack using Queue
⇒ Design Browser history
⇒ Sort stack
...
⇒
tomorrow
⇒ Hash table + sliding window
⇒ Linked list
⇒ Stack and Queue