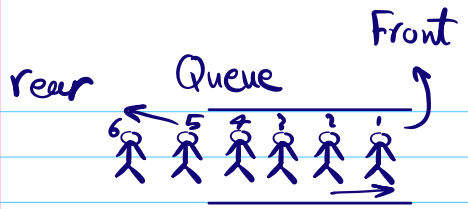
Topics

1 - Qs
2 - Qs with LL
3 - Q with SS
4 - Perfect nums
5 - largest item in window of size B

LIFO
Stack



FIFO
First in First out

Queues Intro.

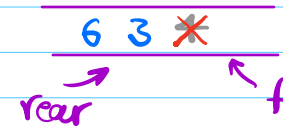
Queue Operations

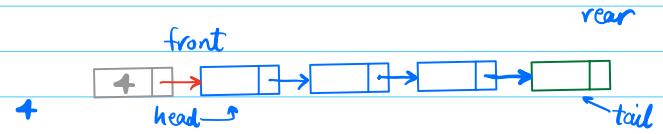
enqueue	Insert a new object from rear
dequeue	Removes & returns the object from front end
isEmpty	whether any item is in Q
Peak front	get the front element
rear	get the rear element

Peak → Similar to stack Peak

P1 Implement queue using Linked List:

enqueue
Q.enq(4)
Q.enq(3)
Q.enq(6)
dequeue
Q.dq()

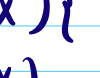





```
void enqueue(int x){
    nn = new node(x)
    if(head == null){          // isEmpty()
        head = nn;
        tail = nn;
    }else{
①       tail.next = nn;
②       tail = nn
    }
}
```

new node

O(1)




```
int dequeue(){
    if(head == null){          // isEmpty()
        error, ex, flag
        ret
    }else{ // Q is not empty
        x = head.data
        head = head.next
        if(head == null){      // isEmpty()
            tail = null
        }
        ret x
    }
    ret x
}
```

O(1)

Q.isEmpty

```
bool isEmpty(){
    ret (head == null)
}
```

O(1)

```
int front(){
    if(isEmpty()) ret ex, error, ...
    else ret head.data
}
```

O(1)

equivalently

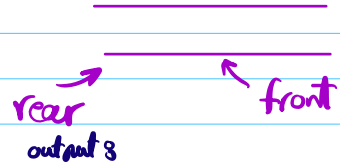
```
int rear(){                    // isEmpty()
    if(tail == null){
        ret ex, error, flag etc
    }else{
        ret tail.data
    }
}
```

O(1)

P2 Implement Queue using 2 stacks.

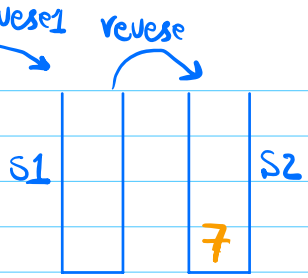
front
rear
HW

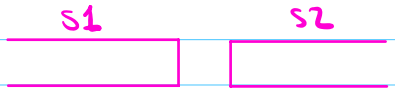
Q.enq(4)
Q.enq(3)
Q.deq()
Q.deq()
Q.enq(6)
Q.deq()



6 3 4
③ ② ①

Q





~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~, ~~6~~, 7

enq(1)
enq(2)
enq(3)
deq() ①
enq(4)
enq(5)
deq() → ②
deq() → ③
deq() → ④
enq(6)
enq(7)
deq( ) → ⑤
deq() → ⑥

```
void enqueue(int x){
    st1.push(X)
}
```

O(1)

```
boo isEmpty(){
    ret (st1.isEmpty() && st2.isEmpty())
}
```

O(1)

tn / n → O(1)

```
int dequeue(){
    if (isEmpty()) ret ex, error, log etc.
    if(st2.isEmpty()){ // refill only if empty
        while(!st1.isEmpty()){
            st2.push(st1.pop())
        }
    }
    ret st2.pop()   O(1)
}   avg O( )
```

O(n)

∀ item average asymptotic analysis

2push+ 2pop for each Item

break ?

Perfect number: any number that is composed of 1 and/or 2

P3 Find n^{th} number with digits only 1 or 2 or both.

1
2

1, 2, 11, 12, 21, 22, 111, 112, 121, 122, ...

1st 2nd 3rd 4 +1 5 +2 6 +1 7 +2 8 9 10

1, 2, 3, ..., 9, 10,
11, 12, 13, 14, ..., 19, 20
21, 22, 23, ... 30
31 ... 40
:
100 101 ... 111 112 113 ...

$\binom{9}{10}$

11211

6th → ans = 22

8th → ans = 112

idea1:

jth list ithNumber

for=1, 2, ..., upper_bound

check if digits are only 1 and/or 2

idea2:

use Q

```
if (nth<=2) ret nth
//init. Q
q.enq(1); q.enq(2)
for( i=3; i<=nth ; i+=2){
    x = q.deq();
    y = 10*x+1
    z = 10*x+2
    if(i==nth) ret y;
    if( i+1==nth) ret z;
    q.enq(y); q.enq(z)
}
```

f

~~1~~, ~~2~~, 11, 12 21 22

smaller nums to large nums

TC: HW

O(n)

↓

nth

$\log_{10}^{x}$

x.tostring()+"1"
+"2"

if x, y, z overflow

deque (deck) pronounced as deque

intro cheque

① add front → ③ removeFront

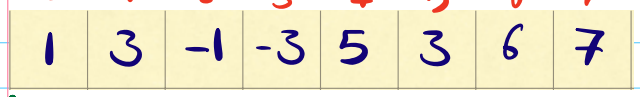
② add back → ④ removeBack

① & ④ → Normal Q

① & ③ → Stack

P4 Given an int array a, find max in all sliding window of size B.
output is an array for windows of size B.

ex



B = 3 $\quad 8-B+1=6$

⊞B $\quad (n-B+1)B$ $\quad$ O(B)

output: {3, 3, 5, 5, 6, 7}

ex

B = 6 $\quad 9-B+1=4$

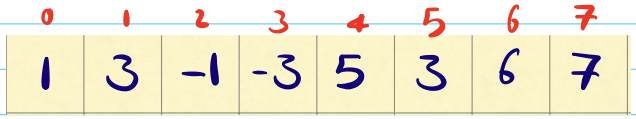
⊞B

output: {7, 7, 7, 7}

idea 1 TC: O(n×B)

idea 2 size PQ keep it B, O(n log B) optional, later in heap
max heap

idea 3

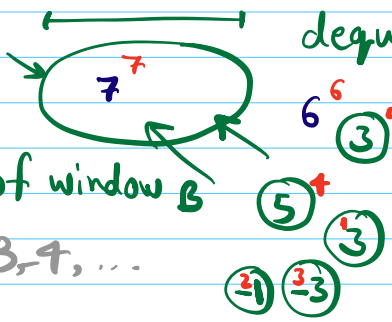


step

start index

A[i]

* front of deque is always max of window



HW

O(N+B)

1, 1, 1, 5, -1, -2, -3, 4, ...

```
// q is my deque
n = A.len
int strart = 0 // start index of window B
for(i=0; i<B; i++){
    while(!q.isEmpty() && A[q.back()] < A[i])
        q.removeBack();
    q.addBack(i);
}
ans.add(A[q.front()]);
for(i=B; i<n; i++){
    //A[i]
    while(!q.isEmpty() && A[q.back()] < A[i]){
        q.removeBack();
    }
    q.addBack(i)
    // remove left most from dequ only if it was
    // affecting max
    if(start > q.front()) q.removeFront();
    ans.add(A[q.front()]);
    start++
}
ret ans.
```

index

index

0 1 2 3 4 5 6 7
1, 1, 1, 1, 5, -3, -6, 3, -100

-100 -101

f b

5 3 -3 -6

5 6

while might be easier to understand