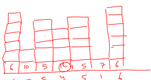
[6, 2, 5, 4, 5, 1, 6].



6 2 5 4 5 1 6
0 1 2 3 4 5 6

i 0 1 2 3 4 5 6
0 6 2 5 4 5 1 6 0
PS[] = -1 -1 1 1 3 -1 5
NS[] = 1 5 3 5 5 7 7
with
area = 1/6 5/10 1/5 3/15 1/5 1/7 1/6

NS[i] - PS[i] - 1

TC = O(n).
AS = O(n).

Queue.

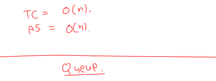
F150

4, 8, 3, 4, 5

class Queue {
int arr[1000], int back, int front;
public:
push () → O(1)
pop () → O(1)
front() → O(1)
size() → O(1)
empty() → O(1)
}

TODO

Push (1)
push (2)
push (3)
front ()
pop()
front()

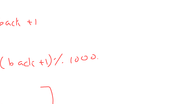


back = back + 1

back = (back + 1) % 1000.

class Queue.
Linkedlist ll;
public:
push() → O(1)
pop() → O(1)
front() → O(1)
}
}

TODO.

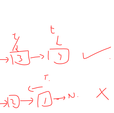


Deque → Doubly-linked Queue

Push/Pop

Queue using 2 stacks

class Queue {
stack S1
stack S2
public:
push () → O(1)
pop () → O(n)
front() → O(n)
!
}



---

s = abcd.

s = a
ab.
abc
abcd

s = aabc

a
aa
aab
aabc

a
#
b
b

→ a#bb

"aabcdcbd"
a#bbbbd#

front = 1.

front
a → λ2
b → λ2
c → λ2
d → λ2

BF: TC: O(N)
AS: O(1)

front
a → λ2
b → λ2
c → λ2
d → λ2

F150
Queue
pop
push

aabcdcbd.
a#bbbbd#

```
class Queue {
    int front, back, size;
    int arr[1000];
    public:
    void push(int x) {
        if (front == -1 && size == 0) {
            front = 0;
            back = 0;
            arr[front] = x;
            size = 1;
            return;
        }
        else if (size == 1000) {
            cout << "Queue is full" << endl;
            return;
        }
        else {
            back = (back + 1) % 1000;
            arr[back] = x;
            size++;
            return;
        }
    }
    int pop() {
        if (size == 0) {
            cout << "Queue is empty" << endl;
            return -1;
        }
        int temp = arr[front];
        front = (front + 1) % 1000;
        size--;
        return temp;
    }
    int front() {
        if (size == 0) {
            cout << "Queue is empty" << endl;
            return -1;
        }
        return arr[front];
    }
    int size() {
        return size;
    }
    bool empty() {
        if (size == 0) {
            return true;
        }
        else {
            return false;
        }
    }
};
```

→ O(n)
→ O(n)
→ O(n)
⇒ TC = O(N)

[12, -1, -7, 8, -15, 30, 16, 28] K = 3.
n = 8

[-1, -1, -7, -15, -15, 0]

BF:
TC: O((n-(n%k))k)
= O(n*k)
AS = O(1)

0 1 2 3 4 5 6 7.
[12, -1, -7, 8, -15, 30, 16, 28]

-1, -7, -15

→ 8, 16, 28 →

[-1, -1, -7, -15, -15, 0]

TC: O(n).
AS: O(K)

[1, 2, 3, 1, 4, 5, 2, 3, 6], K = 3

3
3 4
5
5 5 6

BF: TC: O(n*k)
AS: O(1)

[1 2 3 1 4 5 2 3 2 6], K = 3.

decreasing order

ans. [3, 3, 4, 5, 5, 5, 3]

push_back
pop_front
pop_front