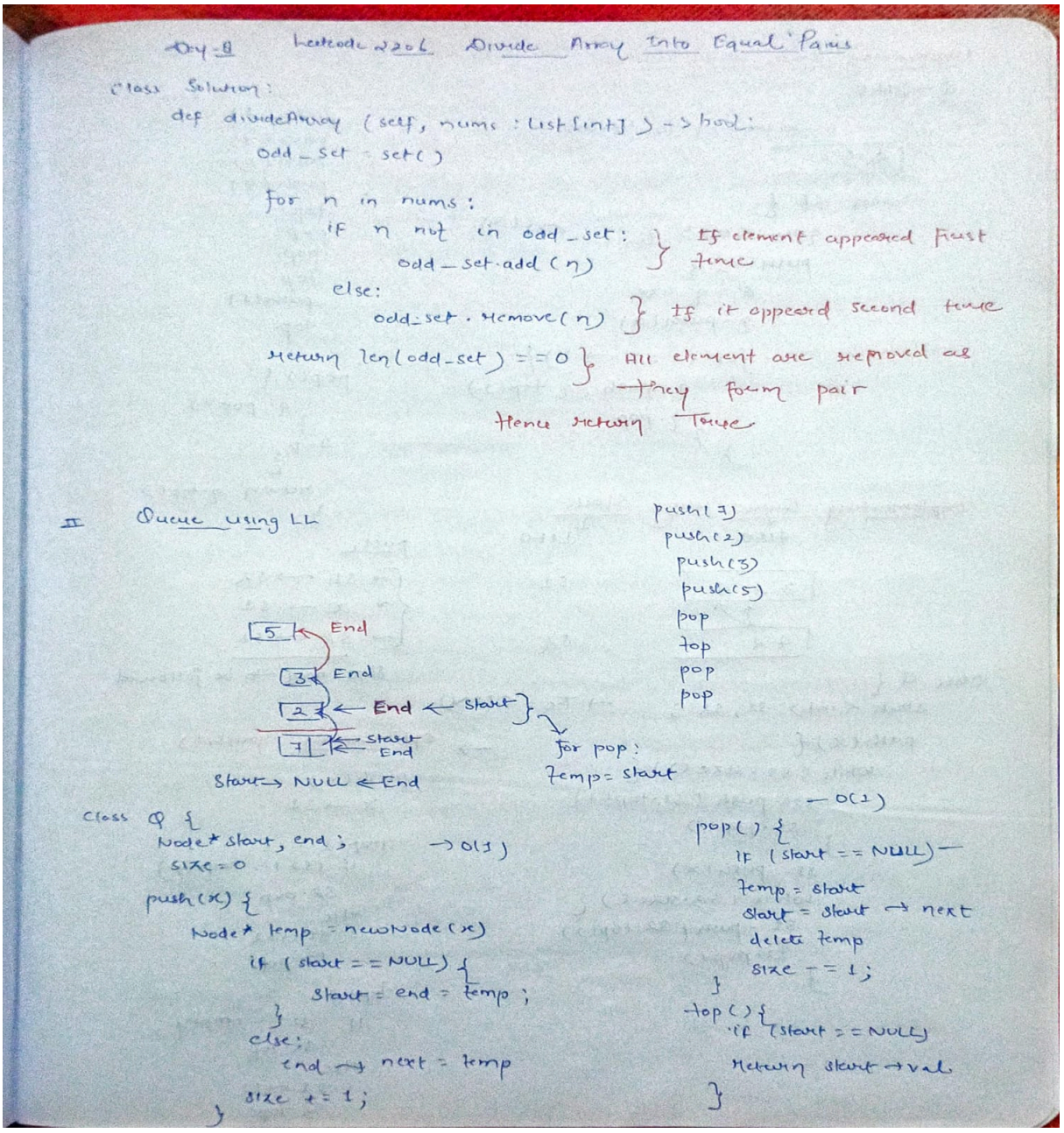
Day_8: DSA



Day 8 leetcode 2206 Divide Array Into Equal Pairs

```
class Solution:
    def divideArray(self, nums: List[int]) -> bool:
        odd_set = set()
        for n in nums:
            if n not in odd_set:
                odd_set.add(n)
            else:
                odd_set.remove(n)
        return len(odd_set) == 0
```

If element appeared first time

If it appeared second time

All element are removed as they form pair

Hence return True

II Queue using LL

Start → NULL ← End

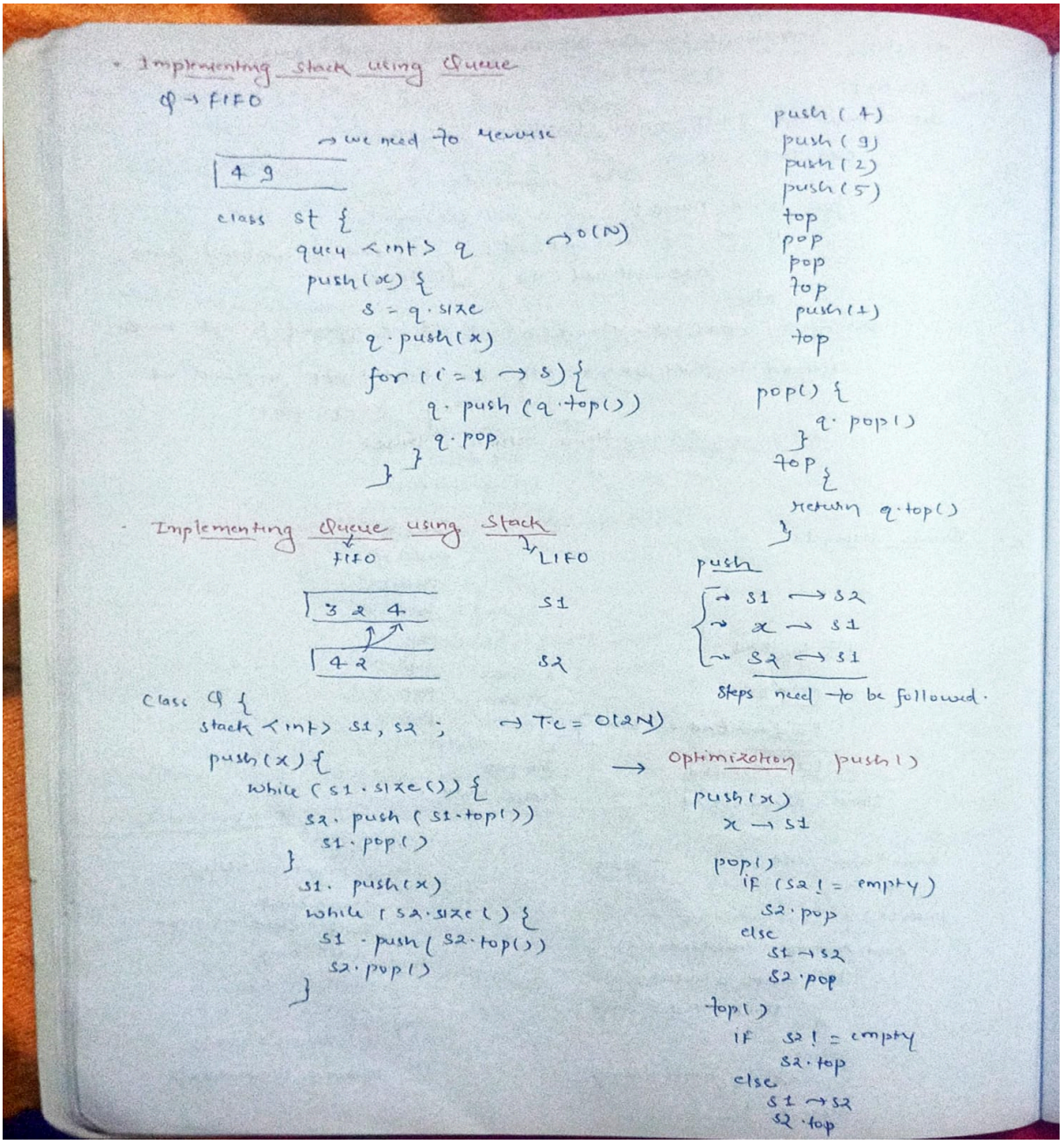
```
push(7)
push(2)
push(3)
push(5)
pop
top
pop
pop
```

for pop:
temp = start

```
class Q {
    Node* start, end;        -> O(1)
    size = 0
push(x) {
    Node* temp = newNode(x)
    if (start == NULL) {
        start = end = temp;
    }
    else:
        end -> next = temp
    size += 1;
}
```

— O(1)

```
pop() {
    if (start == NULL) —
    temp = start
    start = start -> next
    delete temp
    size -= 1;
}
top() {
    if (start == NULL)
    Return start -> val
}
```



- Implementing stack using Queue

q → FIFO

→ we need to reverse

| 4 9 |

class st {
  queue<int> q → O(N)
  push(x) {
    s = q.size
    q.push(x)
    for (i = 1 → s) {
      q.push(q.top())
      q.pop
    }
  }

push(4)
push(9)
push(2)
push(5)
top
pop
pop
top
push(1)
top

pop() {
  q.pop()
}
top {
  return q.top()
}

- Implementing Queue using Stack

FIFO LIFO

| 3 2 4 | s1

| 4 2 | s2

push
- s1 → s2
- x → s1
- s2 → s1

Steps need to be followed.

Class Q {
  stack<int> s1, s2 ; → T.C = O(2N)
  push(x) {
    while (s1.size()) {
      s2.push(s1.top())
      s1.pop()
    }
    s1.push(x)
    while (s2.size()) {
      s1.push(s2.top())
      s2.pop()
    }

→ Optimization push()

push(x)
  x → s1

pop()
  if (s2 != empty)
    s2.pop
  else
    s1 → s2
    s2.pop

top()
  if s2 != empty
    s2.top
  else
    s1 → s2
    s2.top