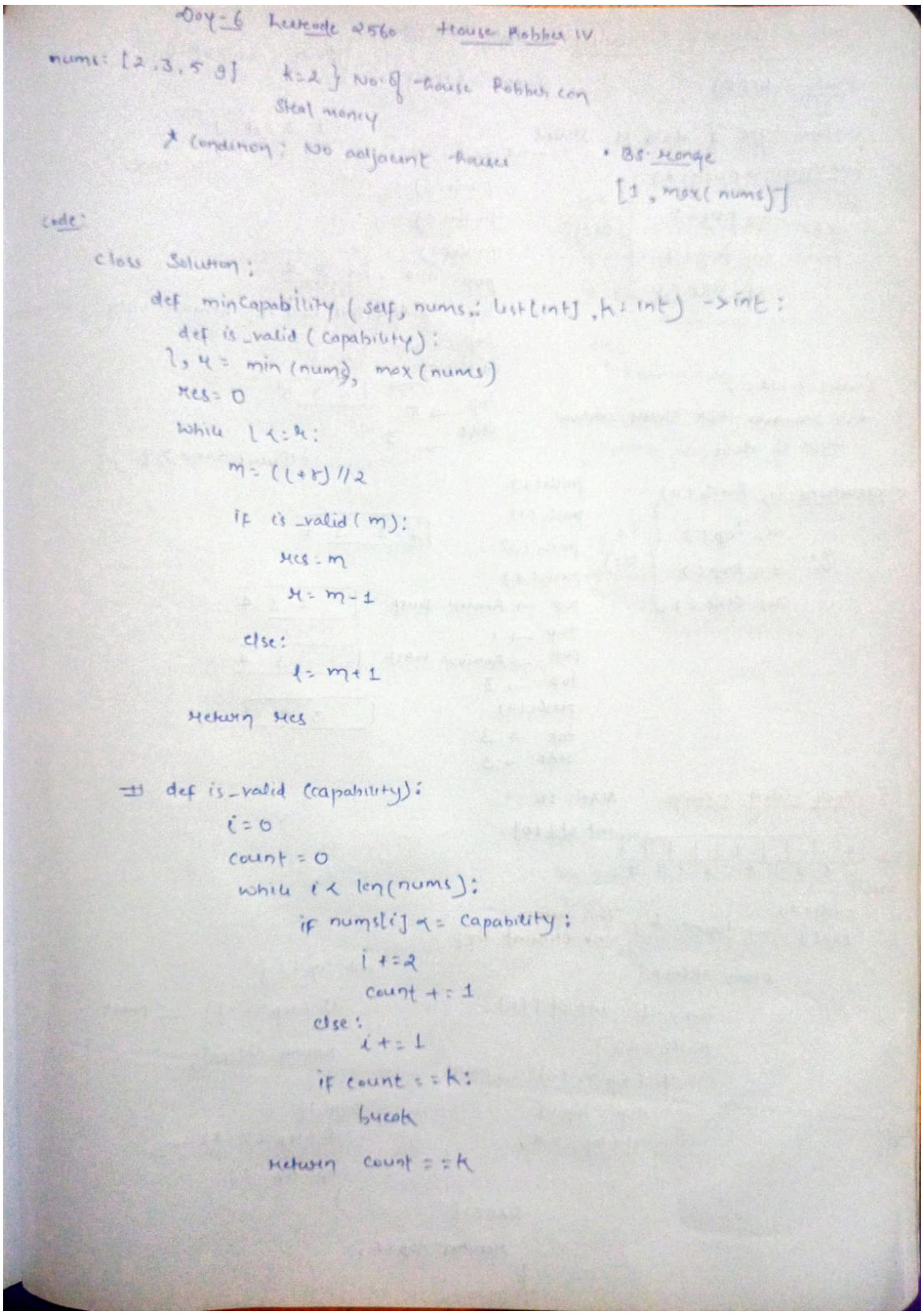
Day_6: DSA



Day-6 Leetcode 2560 House Robber IV

nums: [2,3,5 9] k=2 } No. of house Robber can steal money

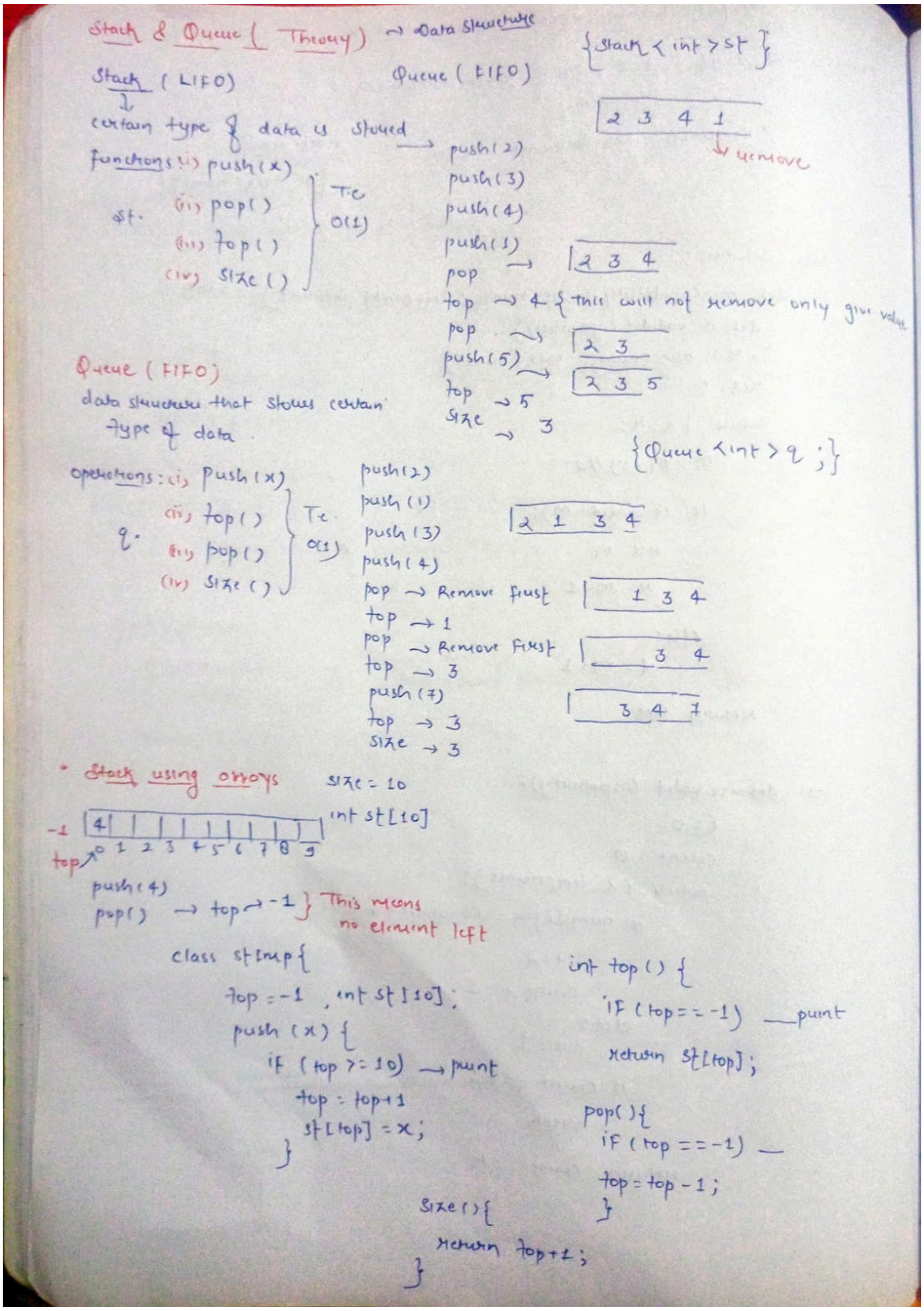
* condition: No adjacent houses

• BS range
[1, max(nums)]

code:

```
class Solution:
    def minCapability (self, nums,: list[int] ,k: int) ->int:
      def is_valid (capability):
      l, r = min (nums), max (nums)
      res= 0
      while l<=r:
          m= ((+r)//2
          if is_valid (m):
              res= m
              r= m-1
          else:
              l= m+1
       return res

# def is_valid (capability):
        i=0
        count = 0
        while i< len(nums):
              if nums[i] <= capability:
                    i +=2
                    count += 1
              else:
                    i += 1
              if count ==k:
                    break
          return count ==k
```



Stack & Queue (Theory) → Data Structure

{ Stack < int > st }

Stack (LIFO)
↓
certain type of data is stored

functions: (i) push(x)
st. (ii) pop()
(iii) top()
(iv) size()
} T.c O(1)

push(2)
push(3)
push(4)
push(1) → | 2 3 4 1 | ↓ remove
pop → | 2 3 4
top → 4 { this will not remove only give value
pop → | 2 3
push(5) → | 2 3 5
top → 5
size → 3

Queue (FIFO)

data structure that stores certain type of data.

{ Queue < int > q ; }

operations: (i) push(x)
q. (ii) top()
(iii) pop()
(iv) size()
} T.c O(1)

push(2)
push(1)
push(3)
push(4) | 2 1 3 4
pop → Remove first | 1 3 4
top → 1
pop → Remove first | 3 4
top → 3
push(7) | 3 4 7
top → 3
size → 3

• Stack using arrays   size = 10

-1 | 4 | | | | | | | | | | int st[10]
top ↗ 0 1 2 3 4 5 6 7 8 9

push(4)
pop() → top → -1 } This means no element left

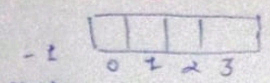
```
class stImp{
    top = -1 , int st[10];
    push (x) {
        if ( top >= 10) → print
        top = top+1
        st[top] = x;
    }

int top () {
    if (top == -1) __ print
    return st[top];

pop(){
    if (top == -1) __
    top = top - 1;
}

size(){
    return top+1;
}
```

Queue using Arrays



Start
End

Size = 4
int Q [4]

cur Size = 0

```
class Q {
    size = 10, q [size] , cur size = 0
    start = -1 , end = -1

    push(x) {
        if (cursize == size) ___ print
        if (cursize == 0)
            { start = 0 , end = 0
        else
            end = (end + 1) % size ;

        q [end] = x , cur size += 1
    }
  top () {
    if (cursize == 0) —
    return q [start] ;
  }
```

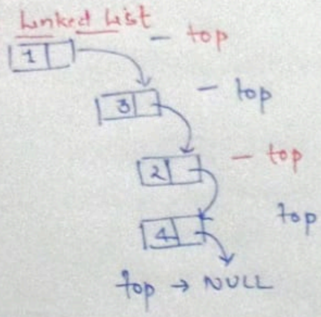
```
pop () {          → Empty Queue
  if (cursize == 0) ___ return
  if (cursize == 1)
        Start = end = -1
  else
     Start = (Start + 1) % size ;

  cursize -= 1 ;
  return el ;
}

size () {
 return Cursize ;
```

• Stack using Linked List

push(4) →
push (2)
push (3)
push (1)
top
pop
push (7)
size



```
→ pop ()

  temp = top
  top = top → next
  delete temp
```

```
class S·T {
    Node* top ; size = 0
  push(x) {
    Node* temp = new Node (x)
     temp → next = top
     top = temp;
     size = size + 1;

top() {
 return top-data
}
```

```
pop() {
 Node* temp = top
 top = top → next
 delete temp
  size = size - 1 } ;
}
          size() {
            return size,
          }
```