**

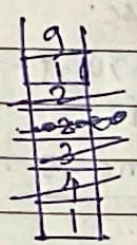
Problem 25: Remove K-digits

DATE

Given non-negative integer num represented as a string, remove k-digits from the number so that the new number is the smallest possible.

Ex:

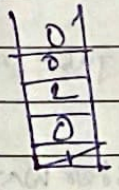
num = "1432219", k = 3



=> "1219"

Ex: num = "10200" k = 1

if top(stack) > the element you are going to insert:
    stack.pop().

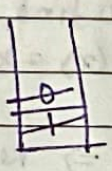


=> "0200"

↳ do left-shift with "0".

Ex:

num = "10" k = 2



if len(num) == k:
    you need to remove all
    Just return "0".

```
def removeKdigits(num, k):
    if len(num) == k:
        return "0"
    for i in num:
        while k > 0 and stack and stack[-1] > i:
            stack.pop(-1)
            k -= 1
        stack.append(i)
    res = "".join(stack[0: len(stack) - k]).lstrip("0")
    if len(res):
        return res
    return -1
```

Problem 26: odd even linked list

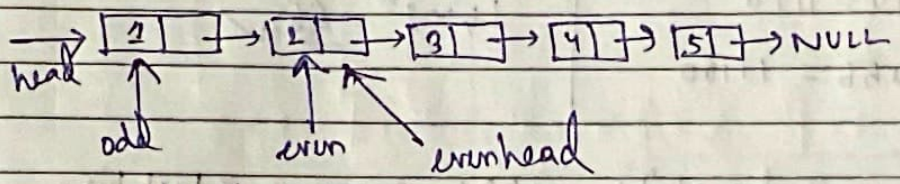
DATE

Input: 1→2→3→4→5→NULL

output: 1→3→5→2→4→NULL

Input: 2→1→3→5→6→4→7→NULL

output: 2→3→6→7→1→5→4→NULL



evenhead = 2 → 4 → NULL

~~head head.next = evenhead~~

head → 1 → 3 → 5 → NULL

head.next = evenhead.

head → 1 → 3 → 5 → 2 → 4 → NULL

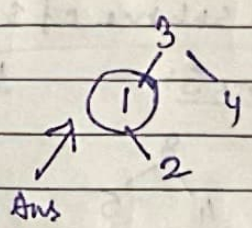
```
def oddEvenList(self, head):
    if head is None or head.next is None:
        return head
    odd = head
    even = head.next
    evenhead = even
    while even is not None and even.next is not None:
        odd.next = odd.next.next
        even.next = even.next.next
        odd = odd.next
        even = even.next
    odd.next = evenhead
    return head.
```

Problem 2: Kth Smallest Element in BST DATE

Given a Binary search tree, write a function kthsmallest to find the kth smallest element in it.

Input: [3, 1, 4, null, 2] k = 1

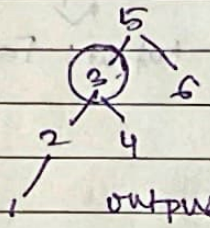
output = 1



Ans

Input:

root = [5, 3, 6, 2, 4, null, null, 1] , k = 3



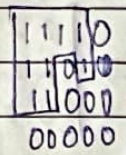
output = 3. ✓

we can do inorder traversal to get ~~elements~~ tree element in sorted order.

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def kthSmallest(self, root: TreeNode, k: int):
        root.arr = []
        if root is not None:
            self.util(root, root.arr)
        else
            return
        for i in range(len(root.arr)):
            if i == k-1:
                return root.arr[i]
    def util(self, root, arr):
        if root is not None:
            self.util(root.left, arr)
            arr.append(root.val)
            self.util(root.right, arr)
        return
```

Problem 29: Number of Islands

DATE



output 1

```
11000
11000
00100
00011
```

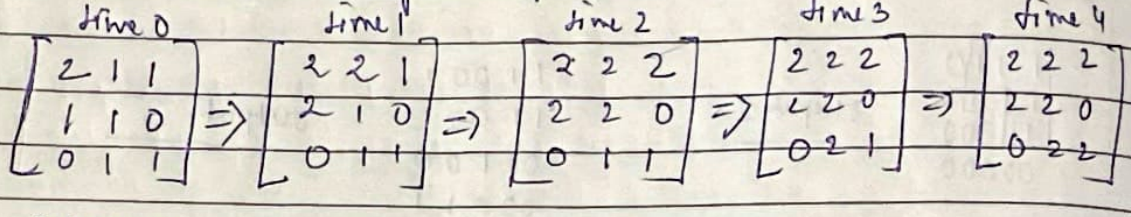
output 3

we can use BFS or DFS to solve the problem.

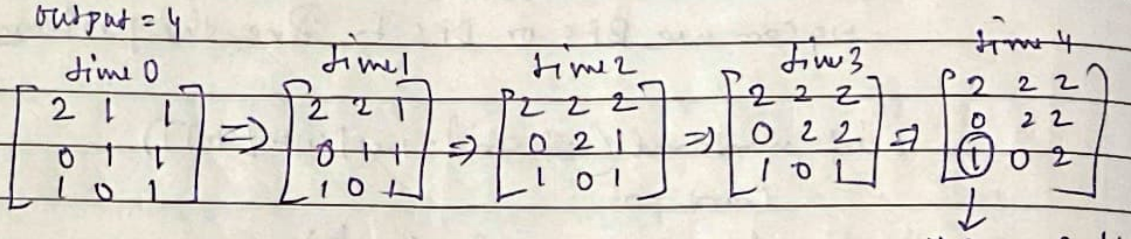
```
def numIsland(self, grid: List[List[str]])
    count = 0
    for i in range(len(grid)):
        for j in range(len(grid[i])):
            if grid[i][j] == "1":
                count += 1
                self.doBFS(grid, i, j)
    return count

def doBFS(self, grid, i, j):
    if i<0 or i >= len(grid) or j<0 or j >= len(grid[i])
        or grid[i][j] == "0";
        return

    else:
        grid[i][j] = '0'
        self.doBFS(grid, i+1, j)
        self.doBFS(grid, i-1, j)
        self.doBFS(grid, i, j+1)
        self.doBFS(grid, i, j-1)
```

DATE

Problem 30: Rotting Oranges



output = 4

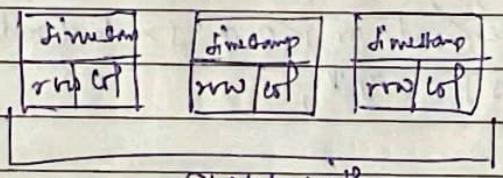


~~this one can't be rotten, so, it will return~~ −1.

[0 2] ⇒ output = 0

we will use queue to achieve BFS on the matrix.

Step 1:



Step 2: pop one by one and insert its neighbours.

```
for i in range(len(grid)):
    for j in range(len(grid[0])):
        if grid[i][j] == 2:
            queue.append(self.Node(0,i,j))
        elif grid[i][j] == 1:
            fresh += 1
while queue:
    x, y, z = i.row, i.col, i.timestamp
    for dx,dy in [(-1,0),(1,0),(0,-1),(0,1)]:
        if 0<=x+dx<rw and 0<=y+dy<cl and
            grid[x+dx][y+dy] == 1:
            grid[x+dx][y+dy] = 2
            queue.append(self.Node(zH, x+dx, y+dy))
            fresh -= 1
def queue[0]
ans = i.timestamp
if fresh == 0:
    return ans
return -1
```

classmate

PAGE

DATE

Problem 31: Interval List Intersections.

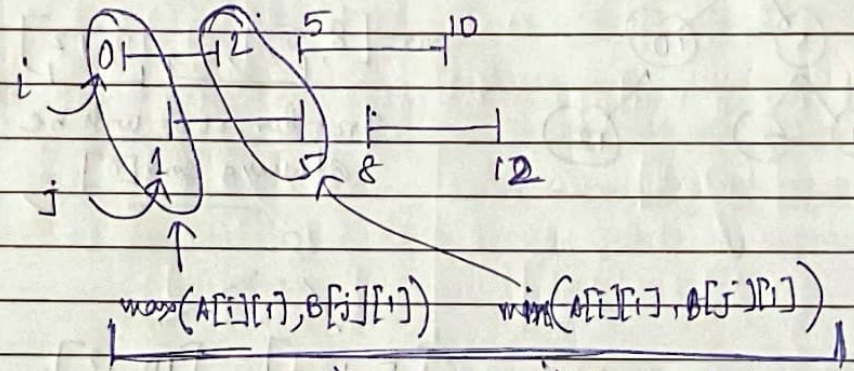
Input:

A = [[0,2],[5,10],[13,23],[24,25]]

B = [[1,5],[8,12],[15,24],[25,26]]

output:

[[1,2],[5,5],[8,10],[15,23],[24,24],[25,25]]



this range will be overlap.

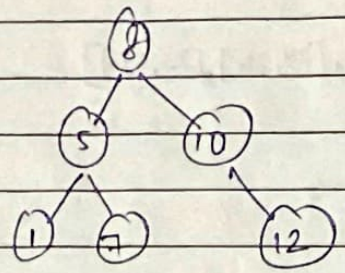
```
if A[i][1] < B[j][1]:
    i++
else
    j++
```

```
while i < len(A) & j < len(B):
    low = max(A[i][0], B[j][0])
    high = min(A[i][1], B[j][1])
    if low ≤ high:
        res.append([low, high])
    if A[i][1] < B[j][1]:
        i += 1
    else j += 1
```

DATE

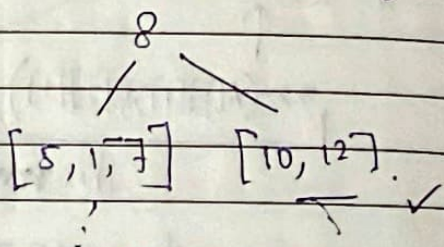
Problem 32: Construct BST from preorder Traversal

Input: [8, 5, 1, 7, 10, 12]
output: [8, 5, 10, 1, 7, null, 12]



arr = [8, 5, 1, 7, 10, 12]

preorder, root will be always arr[0] ✓



recursively we can build the tree. ✓

```
def bstFromPreorder(self, preorder):
    root = None
    n = len(preorder)
    if n > 0:
        i = 0
        root = TreeNode(preorder[0])
        while i < n:
            if preorder[i] > root.val:
                break
            i += 1
        root.left = self.bstFromPreorder(preorder[1:i])
        root.right = self.bstFromPreorder(preorder[i:])
    return root.
```

Problem 33: Uncrossed Lines

DATE

A = 1 4 2
B = 1 2 4
output = 1

A = [2, 5, 1, 2, 5]
B = [10, 5, 2, 1, 5, 2]
output = 3 ✓

		1	2	4
	0	1	2	3
1	0	1	1	1
4	1	1	1	2
2	2	1	2	2

Ans

if matches ⇒ get diagonal value
else max(dp[i-1][j], dp[i][j-1]) ✓

```
dp = [[0 for j in range(len(B)+1)] for i in range(len(A)+1)]
for i in range(1, len(A)+1):
    for j in range(1, len(B)+1):
        if A[i-1] == B[j-1]:
            dp[i][j] = 1 + dp[i-1][j-1]
        else:
            dp[i][j] = max(dp[i-1][j], dp[i][j-1])
return dp[len(A)][len(B)]
```
✓

DATE

Problem 34: Largest Divisible Subset:

Given a set of distinct positive integers, find the largest subset such that every pair (S_i, S_j) of elements in this subset satisfies:

$$S_i \% S_j == 0 \text{ or } S_j \% S_i == 0$$

If there are multiple solutions, return any subset is fine.

Ex: [1, 2, 3] → [1, 2] [1, 3]

[1, 2, 4, 8] → [1, 2, 4, 8]

[1, 2, 3, 4, 6, 10, 8, 24]

[1, 2, 4, 8, 24]

Step 1: Sort the array [1, 2, 3, 4, 6, 8, 10, 24]

Step 2: dp = [[1], [2], [3], [4], [6], [8], [10], [24]]

Step 3: dp = [[1], [2], [3]]

i = 1 j = 0

[[1], [1, 2], [3]]

i = 2 j = 0

[[1], [1, 2], [1, 3]]

[June Day 13]

DATE

Problem 35: Permutation Sequence.

The set [1, 2, 3 ... n] contains a total of n! unique permutations.

By listing & labelling all of the permutations in order, we get the following sequence for n = 3

"123"
"132"
"213"
"231"
"312"
"321"

given n & k, return the k^{th} permutation.

Input n = 3, k = 3	Input: n = 4 k = 9
output "213"	output: "2314"

n will be between 1 & 9 inclusive
k will be between 1 & n! inclusive.

for example: n = 3

first two permutation 123, 132 starts with 1 & they are total count (n-1)!

the next two permutation 213, 231 → starts with 2

so, for 1st digit index will be ceil(k/(n-1)!)

now remove this index character from the list.

also, reduce k values.

```
a = [i for i in range(1, n+1)]
res = ""
perm = 1
for i in range(1, n+1):
    perm *= i
j = 0
while j < n:
    perm //= n-j
    index = (k-1) // perm
    val = a.pop(index)
    res += str(val)
    k -= index * perm
    j += 1
return res.
```

✓

DATE

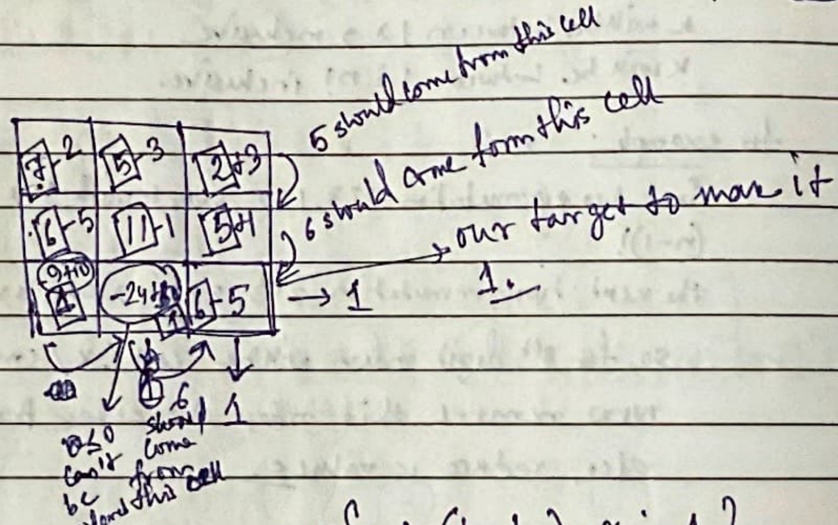
Problem 36: Dungeon Game

-2(K)	-3	3
-5	-10	1
10	30	-5(P)

Knight (K)
princess (P)

The knight has an initial health point represented by a positive integer. If at any point his health point drops to 0 or below, he dies immediately.

Write a function to determine the knight's minimum initial health so that he is able to rescue the princess.



$$\max\{\min(h_1, h_2) - x; 1\}.$$

```
rows = len(dungeon)
cols = len(dungeon[0])
dp = [[float('inf')]*(cols+1) for _ in range(rows+1)]
dp[rows-1][cols] = dp[rows][cols-1] = 1
for i in range(rows-1, -1, -1):
    for j in range(cols-1, -1, -1):
        dp[i][j] = max(1, min(dp[i+1][j], dp[i][j+1]) -
                                  dungeon[i][j])
return dp[0][0].
```

DATE

Problem 37: Single Number II

Given a non empty array of integers, every element appears three except for once. Find that single one.

[2, 2, 3, 2] output: 3

[0, 1, 0, 1, 0, 1, 99] output: 99

Soln 1:

ones ⇒ to stores the number which presents single time.

tows ⇒ to stores the numbers which present two times.

[2, 2, 2, 3]

ones = ∅	2	0
tows = ∅	∅	2

00	00	10
^10	^10	^10
10	10	00
&11	&01	&11
10	00	00

00	10
^10	^10
10	00
&01	&11
00	10

---

return ones.

```
ones = 0
tows = 0
for i in nums:
    ones = (ones ^ i) & (~tows)
    tows = (tows ^ i) & (~ones)
return ones.
```

Soln 2:

0	0	1	0
0	0	1	0
0	0	1	0
0	0	4	1

→ count no of set bits

do modulo with 3.

0 0 4 1

0 0 1 1 → returns 3. ✓

---

classmate

PAGE

DATE

Problem 38: Unique Binary Search Tree

Given n, how many structurally unique BST's that stores values 1 ... n?

Ex: Input: 3
Output: 5

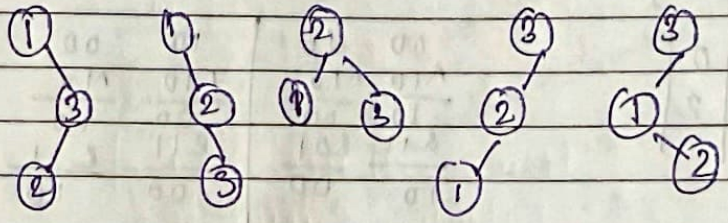
Input 1: [1] ① output = 1

Input 2: [1,2] ① ② ① output = 2

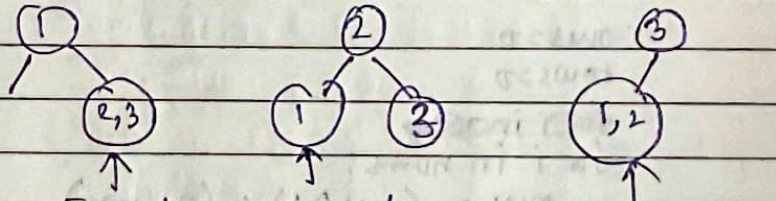
② ①

root as 1 = output(Input 1)
root as 2 = output(Input 1)

Input 3: [1,2,3]

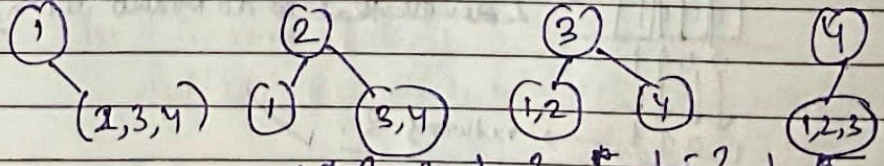


when root = 1 when root = 2 when root = 3



2 + 1 + 2 = 5

Input 4: [1,2,3,4]



5 + 1 * 2 = 2 + 2 * 1 = 2 + 5

= 14. ✓

```
dp = [0]*(n+1)
dp[0] = 1
dp[1] = 1

for i in range(2, n+1):
    for j in range(i):
        dp[i] += dp[j] * dp[i-j-1]
return dp[n]
```

classmate PAGE

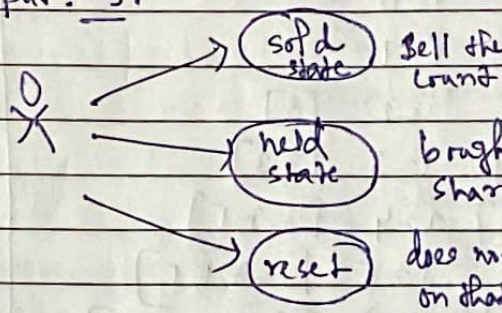
DATE

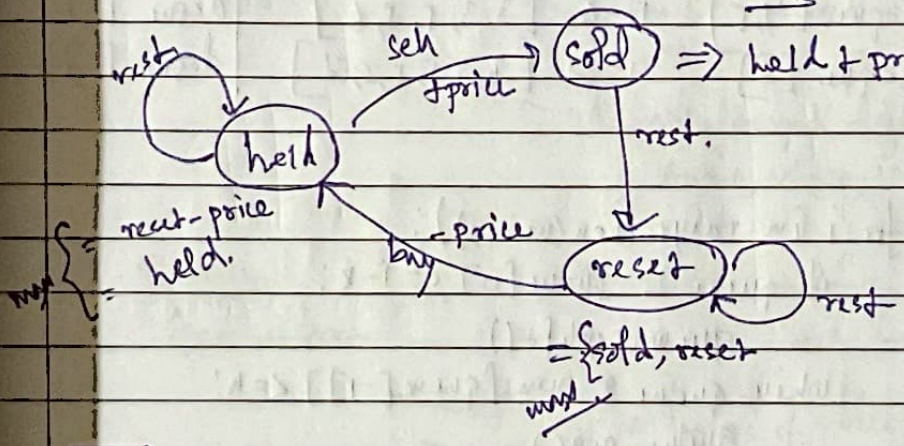
Problem 39: Best time to Buy & Sell stock with cooldown.

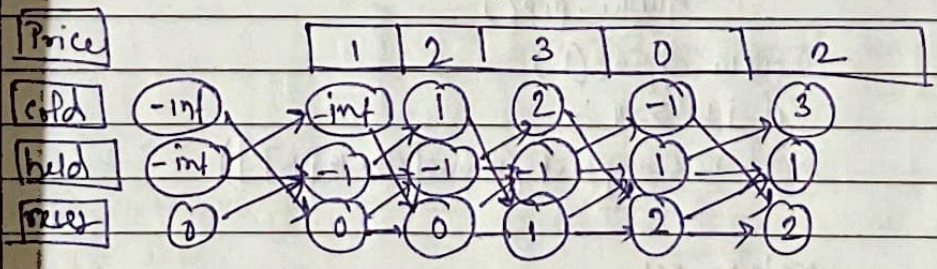
Input: [1, 2, 3, 0, 2]

transactions = [buy, sell, Cooldown, buy, sell]

output: 3.








```
sold = float(-inf)
held = float(-inf)
reset = 0

for i in range(len(prices)):
    presold = sold
    sold = held + prices[i]
    held = max(held, reset - prices[i])
    reset = max(reset, presold)
return max(sold, reset).
```

DATE

Problem 40: Sliding window Maximum

[1, 3, -1, -3, 5, 3, 6, 7]

deque[ ]	i=0		deque[0]
deque [0]	i=1		deque [ ]
deque [1]	i=2	[3]	deque [1]
deque [1,2]	i=3	[3,3]	deque [1,2]
deque [2,3]	i=4	[3,3,5]	deque [ ]
deque [4]	i=5	[3,3,5,5]	deque [4]
deque [4,5]	i=6	[3,3,5,5,6]	deque [ ]
deque [6]	i=7	[3,3,5,5,6,7]	deque [ ]

```
for i in range(len(nums)):
    if queue & queue[0] <= i-k:
        queue.popleft()
    while queue & nums[queue[-1]] <= n:
        queue.pop()
    queue.append(i)
    if i+1 >= k:
        res.append(nums[queue[0]])

return res.
```