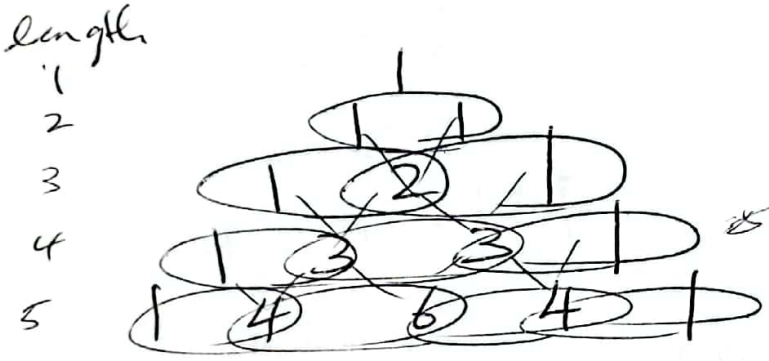
5.20

Input: integer ; Output: input number of of lists

Edge Case: if r=1 , return 1

Constraints: r > 1 and r ∈ ℤ

The left and right most sides are 1 always
There needs to be an equation to get the item within left and right most.



Output = [[1], [1, 2, 1] , [1, 3, 3, 1] , [1, 4, 6, 4, 1]]
0 1 2 0 1 2 3

We can use a for-loop foreach row before the row we are calculating.

result = [ ], new_row = [1]
for i in range of ~~prev row length~~ input starting will 1
~~new row~~ = ~~new row~~
~~new row~~
~~R~~[i] = result[-1][i-1] + result[-1][i]

new_row.append(1)
result.append(new_row)

This will take O(n) space and $O\left(\frac{n(n+1)}{2}\right) \approx O(n^2)$ time

Ex input : r = 4

~~result = [ ] , new_row = [1]~~

~~# for loop~~

~~if~~

We need a while-loop first and increment up to the input r.

index = 1 — result = [[1], [1,1]]

while 1 ≤ 4

new_row = [1]

for loop

new_row[1] = result[-1][0] + result[-1][1] i = 1

= 1 + 1

= 2

result = [[1], [1,1], [1,2,1]]

for-loop

new_row[1] = 1 + 2 = 3 i = 1

new_row[2] = 2 + 1 = 3 i = 2

result = [[1], [1,1], [1,2,1], [1,3,3,1]]

2/4

```
def fun(r:int) -> List[List[int]]:
    result = [[1], [1,1]]
    if r == 1:
        return [[1]]

    if r == 2
        return result

    if r == 0
        return []

    index = 1
    while index < r:
        new_row = [1]
        for i in range(len(result[-1])):
            new_row[i] = result[-1][i-1] + result[-1][i]

        new_row.append(1)
        result.append(new_row)
        index += 1
    return result.
```

24 min
40 min to write code
50 min to pass all tests
3/4

Evaluation:

Took too long trying figure out that I had to append to new_row

↳ `new_row[i] = ~~` X
`new_row.append(~~)` ✓

The index had to start from the length of the already existing list (result = [[1], [1,1]])

↳ Next time pay attention to the base case and keep track.

4/4

6/28/22

5.9

Input: $n \in \mathbb{Z}$; Output: List[int] of primes

Constraints:

- Input is $n \in \mathbb{Z}$ where $n > 1$
- The items in the output list are primes so they do not have divisors other than 1 and itself.

How to check or generate primes?

- Naive (Brute Force) approach is to generate a uniform iterating list of numbers less than the input and del items that is divisible by other items in the list.

The first loop will start from the end of list and second ~~will~~ from start

Ex ~~[1, 2, 3, 4, 5]~~

~~[5, 4, 3, 2, 1]~~

[1, 2, 3, 4, 5]

This will take $O(n^2)$ and $O(n)$.

1/3

- Can I use pointers?

I ↳ ~~Solve~~ Execute same approach as the naive sol'n and make a new list

Note: I can use `l = list(range(n))`

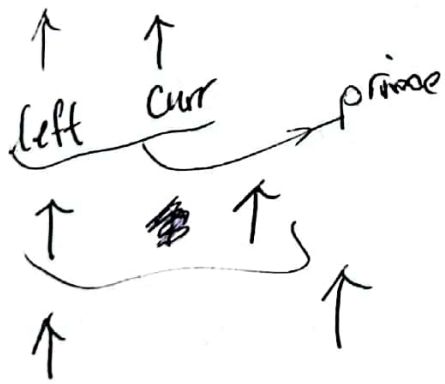
II Or I can use an for loop, ~~and~~ check if prime, and append to empty list.

Both will be $O(n)$ ~~and $O(n)$~~ time

I will be $O(n)$ space

II will be $O(\#\text{ of primes})$ space

Ex [X, 2, 3, 4, 5, 6, 7]



[2]

[2,3]

[2,3,5]

[2,3,5,7]

2/3

```
def fun(n: int) -> List[int]:
    l = list(range(n))          or n
    for i in range(len(l)):
        if l[i]
while curr < len(A):
    if l[curr] % A[left] == 0
        return
        curr += 1
        left = 0
    else:
        ;
```

3/3

6/28/22

5.5

Input: Sorted array ; Output: count of valid elem.

- Naive, brute force approach is to take an item in the list and compare it with ~~of each~~ other items. Then increment, move on to the next item (and keep count of unique) This will take $O\left(\underbrace{\frac{n(n+1)}{2}}_{\text{Gauss}}\right) = O(n^2)$ time and $O(1)$ space.
- We can use hashmap / hashset to iterate through the list and keep count in the dictionary. and return the number of keys in the dictionary. This will take $O(n)$ time and space
- Another approach is to use pointers and check if the subsequent item is same or greater (since the input list is sorted) and keep count using conditionals. This will take $O(n)$ time and $O(1)$ space

Example

starting @ count = 1

[2, 3, 5, 5, 7, 11, 11, 11, 13]

c n r

count ++

count ++

We want to switch places w/ another item if same.

↳ switch w/ current or next?

↳ switch w/ next

2, 3, 5, 13, 7, 11, 11, 11, 5

c n r

count = 4

count = 5

count = 6

We are gonna be in inf. loop if we don't change value of right

2 3 5 13 7 11 11 0 0

c n r

At this point, stop loop

⇒ While curr < right

```
def fun(A: List[int]) -> int:
    count = 1
    curr = 0
    next = 1
    end = len(A) - 1
    while curr < end:
        if A[curr] ≠ A[next]:
            count += 1
            curr += 1
            next += 1
        else:
            A[next] = A[end]
            A[end] = 0
            end -= 1
    return count
```

18 min I remembered the solution write_index = 1

Edge case:

```
if len(A) == 0:
    return 0
```

23 min

Evaluation / correction

- The problem was that I was unable to figure out that I shouldn't have used a while loop.

- Since I am checking two adjacent items in the list, the traversal must be uniform. So use a for loop and compare using i and i+1 or i-1.

- Next time, check if I need to compare adjacent items or items that are someplace else with the current item.

4/4