**What does it mean when it says array problems?**

1. Reverse nodes in k groups here

2.

**What problem can stack solve?**

1. Reverse string here as well

We can also check the length of the dictionary here:

You can do len(dictinoary) here

**Store position and value**1. Can store (value, index) in the stack as you traverse

(this also applies to bfs and dfs)

* Used in path sum here

**Dictionary with a remainder sum**

1. The question in place is make sum divisible by k here

**Stack with a pair**

(index, next greater elem)

Using stack to store a tuple of an index and the next greater element

This is used in the problem next greater frequency element

**How to solve subarray problem**

1. Prefix sum if sum is involved

2. Binary sarch

3. Two pointer can be used too

4. Sliding window

**When to use binary search**

1. We can use bsearch when u can see that if condition(k) is True then condition(k + 1) is True, then we can consider binary search.

2. Can be used for min and max problem

**Tech 4:**

**Using the monotonic stack**

**How to apply this pattern:**

Pattern 1: Find the next immediate greater

problems to discuss

1. 496. Next Greater Element I

2. 1019. Next Greater Node In Linked List

3. 503. Next Greater Element II

Pattern 2: Find the immediate next smaller

1. 1475. Final Prices With a Special Discount in a Shop

Pattern 3: Find the farthest greater

1. 962. Maximum Width Ramp

Pattern 4: Monotonically Decreasing / Inc stack

1. 853. Car Fleet

2. 901. Online Stock Span

3. 769. Max Chunks To Make Sorted

Pattern 5: Lexicographical ordering

1. 402. Remove K Digits

2. 316. Remove Duplicate Letters

**Sliding window trick**

**Technique 1:**

Using map to store the position

1. Keeping the index of the postion and subtracting the distance here, check out the problem contiguous array and then you will know
2. And then when you iteratre through more indexes, you do

maxLength = max(maxLength, i – mp[s])

**Technique 3:**

3 pointer sliding window here

# of nice subarrys leetcode

L, m and right here

1. 3 potiner problem technique here, and then using this we have the code and then

**Kadane’s algorithm**

1. Given an array **arr[]** of size **N**.The task is to find the sum of the contiguous subarray within a **arr[]** with the largest sum.

**Tech 2:**

1. Store the last seen index and the modulus in a dictionary

Dict[remainder] = index

make sum divisible by k here

**Sum of subarry minimums**

Used on pair problem, different paces

**Heap**

1. Top k pattern

2. Merge k sorted pattern

3. Two heap pattern

4. Minimum number pattern

**How to deal with the stuff here**

**1**. And then here we have the code

2.

-

**technique -1:**

**How to sort a map by its frequency in python here**

**-**

**5. Sliding window**

for max and min

Type 1:Dynamic Ex

1. Expand window until we met on the outer loop

while(right < arr.size){

contract window until no longer meet the condition.

while(start < right && curSum >= x)

max = (cur\_max, r- l + 1)

**Shortest subarray greater then number**

Type 2: max consectuive ones

**Trick 6:**

**Binary search application**

1. Can also be used to solve max and min

2. Can be used to solve problem with kth problem

Maximize the minimum difference between any element pair by selecting K elements from given Array

3. And can also work on here

**Using bitwise operator**

1. Can be used to calculate the carry

2. it's logical AND of two input numbers, shifted one bit to the left:

carry = ( x & y ) << 1

3. Using the xor operator can be used to minus number

**Binary search tree tricks here**

1. There are usually 2 ways to solve problem here, one is searching the entire tree based on O(n), the other is using height based on O(n)

2. What happens here when you are trying to find the max or min of sth and you can’t find it by returning recursive value?

Look at the example in closest binary search value, just use a class level variable to solve the problem here

Technique 7:

2 pointer tricks here

1. How to use 2 pointers and make sure no out of bound exception here

while p1 < len(arr1) and p2 < len(arr2) important when using 2 pointer approach

**Trick 1:**

Language feature

Using insert

result.insert(0, value)

this allows you to insert at the front here instead of using a queue

Trick 2 the breakup variable here:

1. The following here

a, b = edge

where the edge can be an array of 2 values array[0] and array[1] here

**Trick 3:**

How to remove a letter from the middle of the word here?

- locatoin = magazine.index(c)

newWord = magazine[:location] + magazine[location + 1:]

So basically the above would work in this case

The other thing you can do is do the followign here

**Trick 5:**

Using ordinance

for i in v:

out = out \* 10 + (ord(i) - ord("0"))

**Trick 6:**

1. How do we turn the number 38 into the following?

38 -> 3+ 8 -> 11

11-> 1 + 1 -> 2

The following can be done using the code in add\_digits:

11-> 1 + 1 -> 2

How to delete an element

l = [1, 4, 6, 2, 6, 1]

print("List before calling del:")

print(l)

del l[3]

Linkedlist section

1.

2.