Problem 1

1 2 3 | 4 | 5 8

Bool isCorrect()

return size = max || size == 1

1 | 3 4

Find (list arr, call time size))

a = split()

if (! is Correct (split))

find();

split()

1 2 3 5 6

if (size ÷ 2)

return fill()

split array in half

if either half is not correct

if both are correct

fill (smaller)

Fill

for i = start + 1; i < end; i++)

retlist.add(i);

return retlist

Boolean is full sequence 이면

if (duplicate)

false

else if

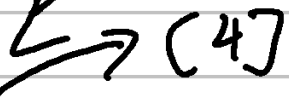
1 2 3 3 5

1 2 3 4 6

size = 5

L R

1 2 | 3 5



3 4 5

6 7 1 2 3

3 = (2) + 1

7 = (2 - 1) + head

(1) → 6

9 10 11 12 14 15 16 17

```
if (sides equal & sequences)
  List.add(end of left)
  list.add (end of right)
  fill(list)
```

Question 2

```
abcd efgh
(har start = str.get(4)
while ( str.get(0) != start)
  advance

advance() // called n times so space is O(n)
  str[0] =(
  for every character
  move left
  add ( to end
```

Question 3

Brute force: get every permutation then check if it's a word. $O(n!)$

Better:

get the set of letters
Search every word. Those w/ all matching letters count, without don't
worst case $O(nm)$ where n is free letters, m is number of words. Probably better than factorial

↳ the same

Solution: Sort all words in lexicographical order, then group all words w/ the same lexicographical signature together

abcd = abcd
dcba = abcd

for each word takes $n(\text{word length})$ passes, then takes m passes through dictionary to search for same signatures $O(nm)$

dcba

Sortalpha()

while (out of order)   adbc

min ascii;

index of min

for every character

if this character is less than the minimum

set counters

else continue

move minimum to the front

Chad

Question 2

Hash table <String, list of string>

if pre-processing allowed

we'd get all into signatures

order the list lexicographically

binary search the list so we don't have to traverse the entire dictionary

abcd

Problem 2: given a file find an integer that appears at least twice

let file be 1,2,2,3,3,4,5,6

Better: using a hashmap:

put everything in terms of

<number, frequency>

Then, search all non-null

O(n)

1 2 3 4

Vector replacement Q3

abcd efgh ghdef abc

a b br

abc defgh

a b br

1) move b br b a then b br a

b br a

same action, done recursively

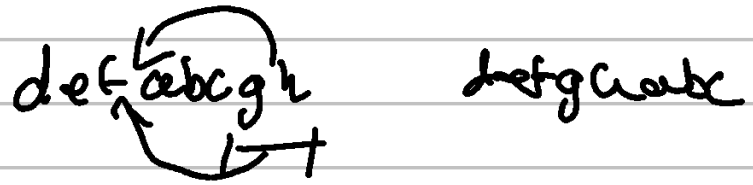
Subtasks

1) get br a

a = substring (0, n) abc

b = substring (n, n+n)

c = b+a



Rotate the Vector abc to be cba

He l lo
A B C

lo He

Rotate (C n_1, n_2)

a = substring(0, n_1)
b = substring(n_1, n_2)
c = substring(n_2)
return cba

Problem 5 push button encoding

Hashmap < Integer signature, Arraylist < string > >

To detect collisions, just check which values have more than one entry

OR: using an array: just store the file in an array, sort it, and every time check the next entry when looking at n_1 and report collision if they are the same