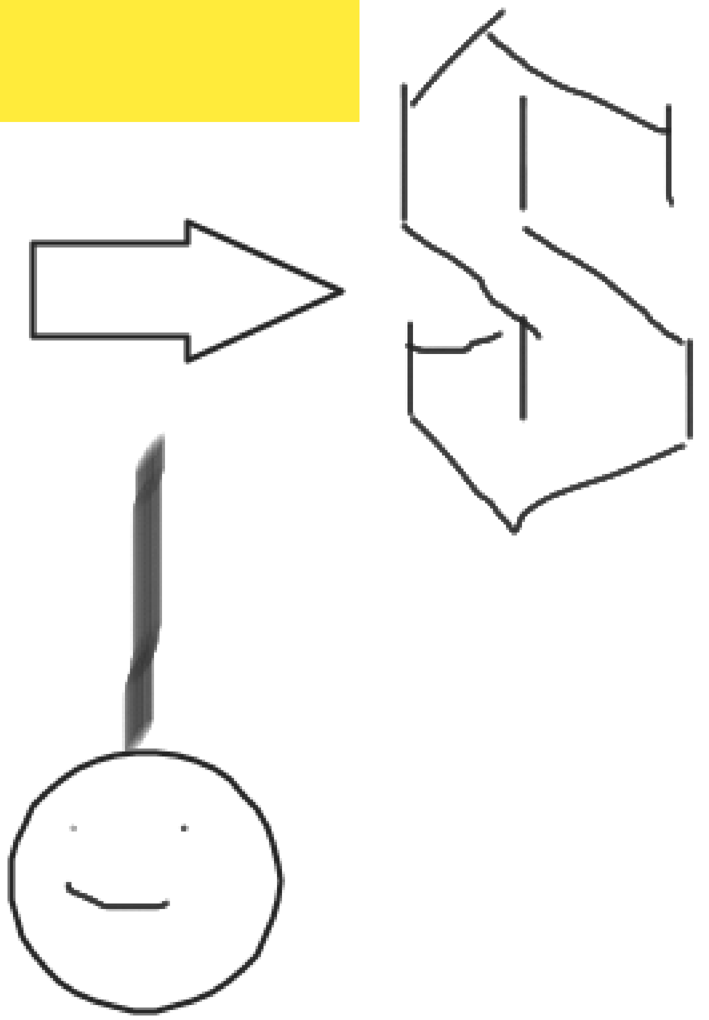
W

Is *QuickSort* always $\mathcal{O}(n\log(n))$? Is *MergeSort* always $\mathcal{O}(n\log(n))$? Justify your answer.
Hint: Consider what this depends on



[0,1,2,3,4,5]
pivot = 0

NO

Mergesort is always O(nlogn) as it roughly splits the list into half n times

-

↑Real

merge sort is always O(nlog(n)) (cuz learned this in lec) and quicksort can be O(n^2) because the paritation has the possiblility to always be the smallest in the list

Mergesort is always nlogn, quicksort depends on ur pivots (worst case n^2)

Quick sort is on average O(n*log(n)) but not always O(n*log(n)), this depends on how pivot is chosen. Merge sort is always O(n*log(n)) because you always split the list in half.

Quicksort worst case: O(n^2)

quicksort can be unevenly split - worst case of O(n^2) fr

Dr. Halstead and Dr. Manning are arguing about the new hospital database system sorting upgrades. Dr. Manning argues Quicksort is the faster option, while Dr. Halstead is a firm believer in the efficiency mergesort. With the help of April, Dr. Choi runs some tests with both sorting algorithms on the hospital dataset. Here is what you know about the dataset:

1. 99% of the time, the hospitals dataset is mostly sorted
2. All the sorting algorithms are run in their non-in-place variants
3. Every other part of their software is as optimized as it can get - T[...]rce of slowdowns other than the choice of algorithm

Note: Since these are doctors and not computer scientists, they do not[...]rt (which would've been the ideal compromise for Will and Natalie)

Given what we know about G[...]abase, what did Dr. [...]n [...]r tests?

[...]tify your answer

bogosort is the way

let \varepsilon > 0

just use stalinsort O(1) 100%

Merge sort is W

Quicksort be the play because u can pick a pivot in the middle and sort the list in fewer calls than compared to mergesort where you always split list into chunks sized-2

bubblesort tho?

quick sort >>>

quicksort has better constants than mergesort, and since the database is mostly sorted, then using the middle pivot would almost always be efficient

quick sort should be better, because the mostly sorted nature of the input should speed up the partition operation, especially if we select the middle element as the pivot

Since the list is sorted roughly, depending on the choice of pivot, I think quick sort would be slower than merge sort most definitely!

no school on friday!!!!!

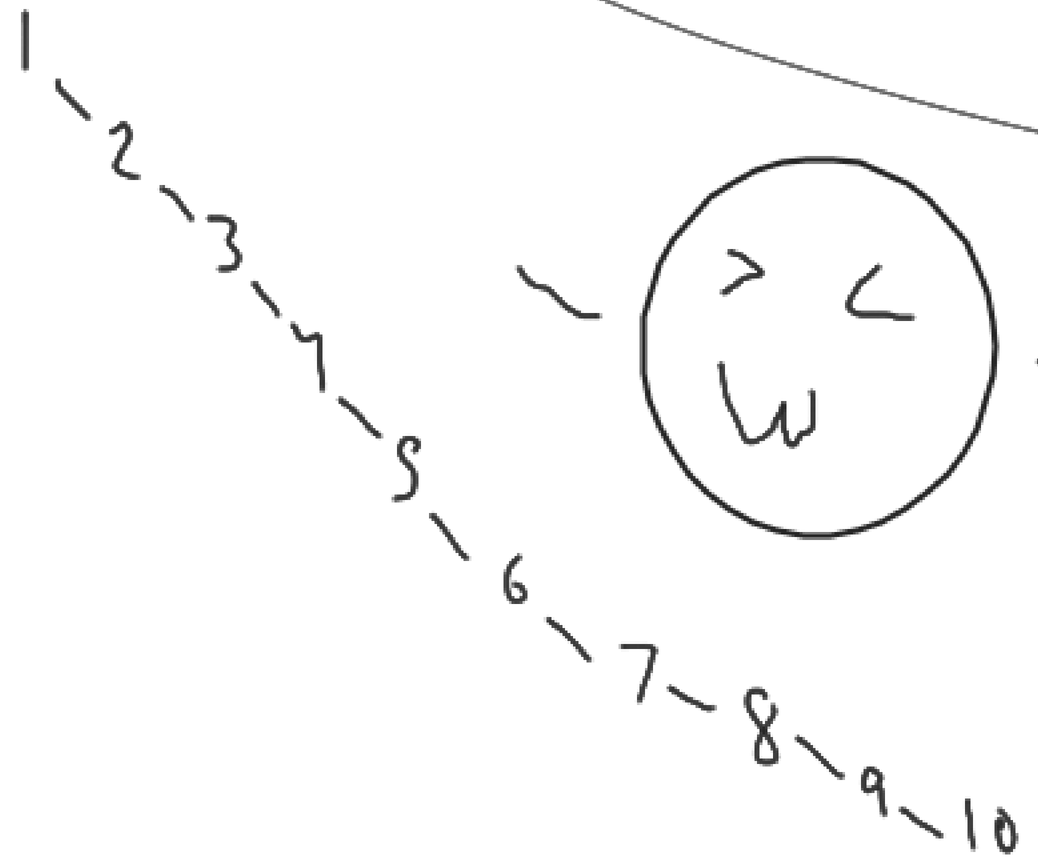
T1: O(n)

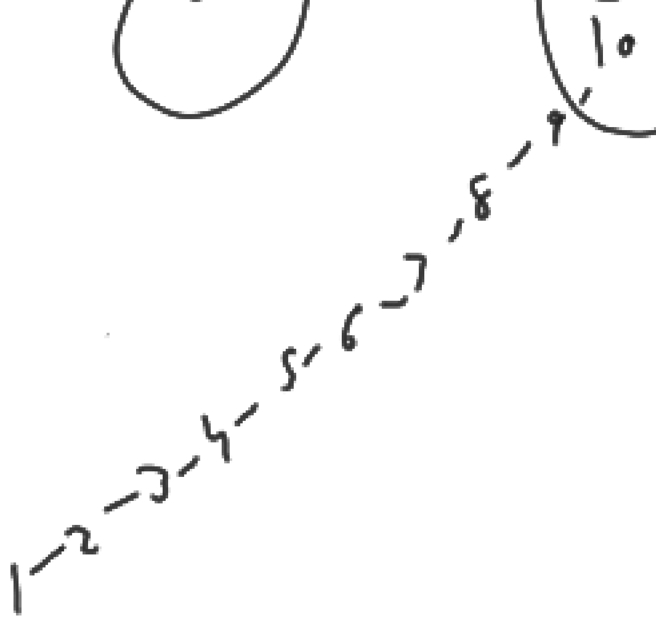
T2: O(n)

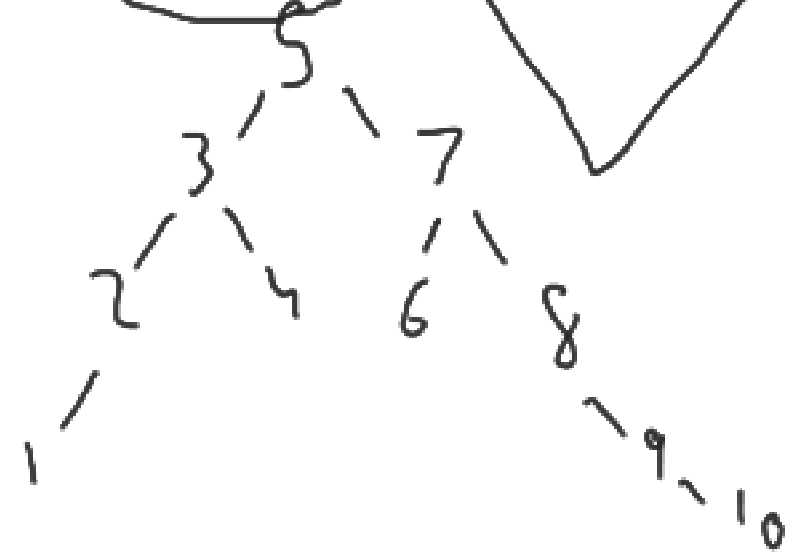
T3: O(log n)

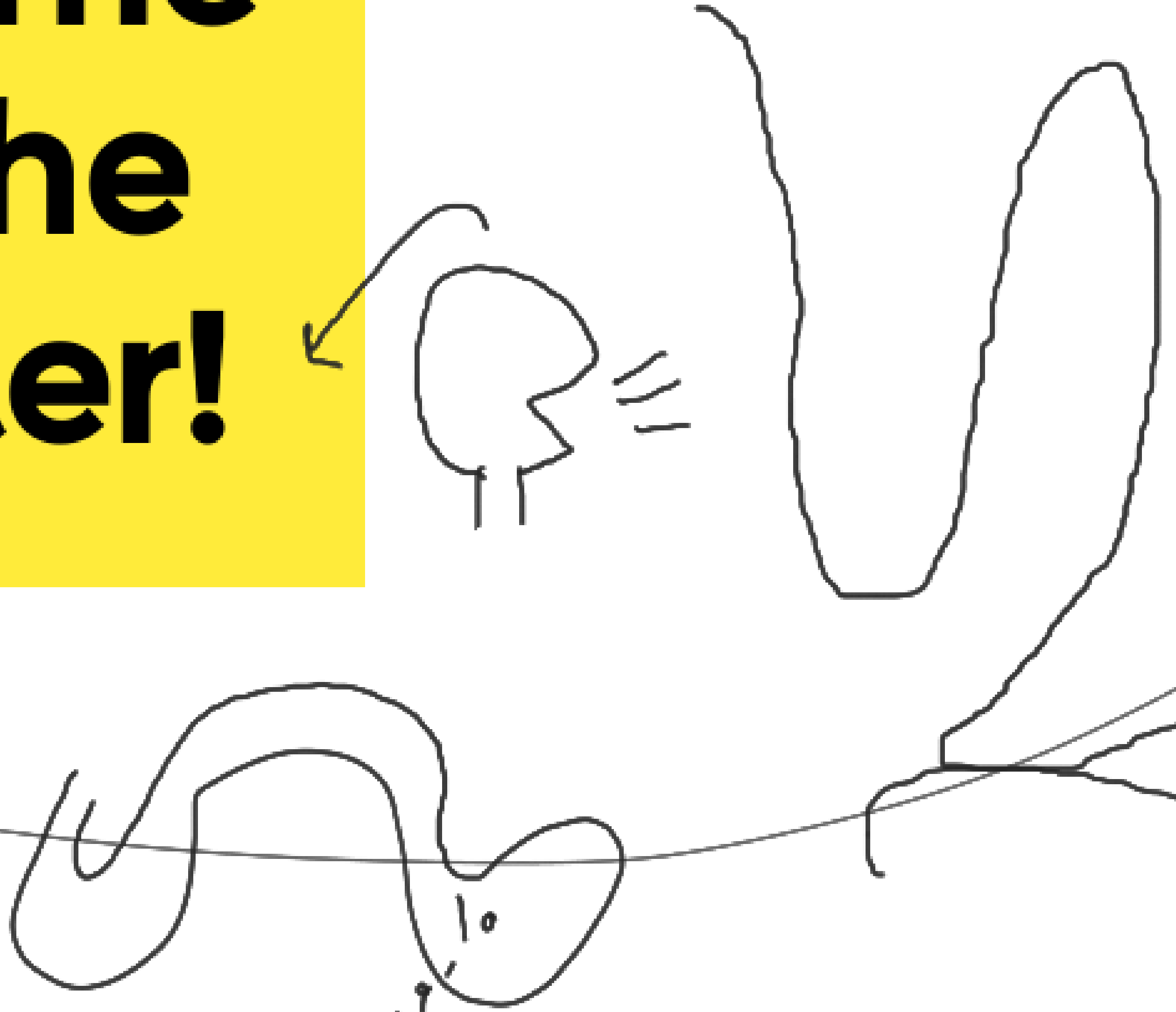
Shuffle is the faster!

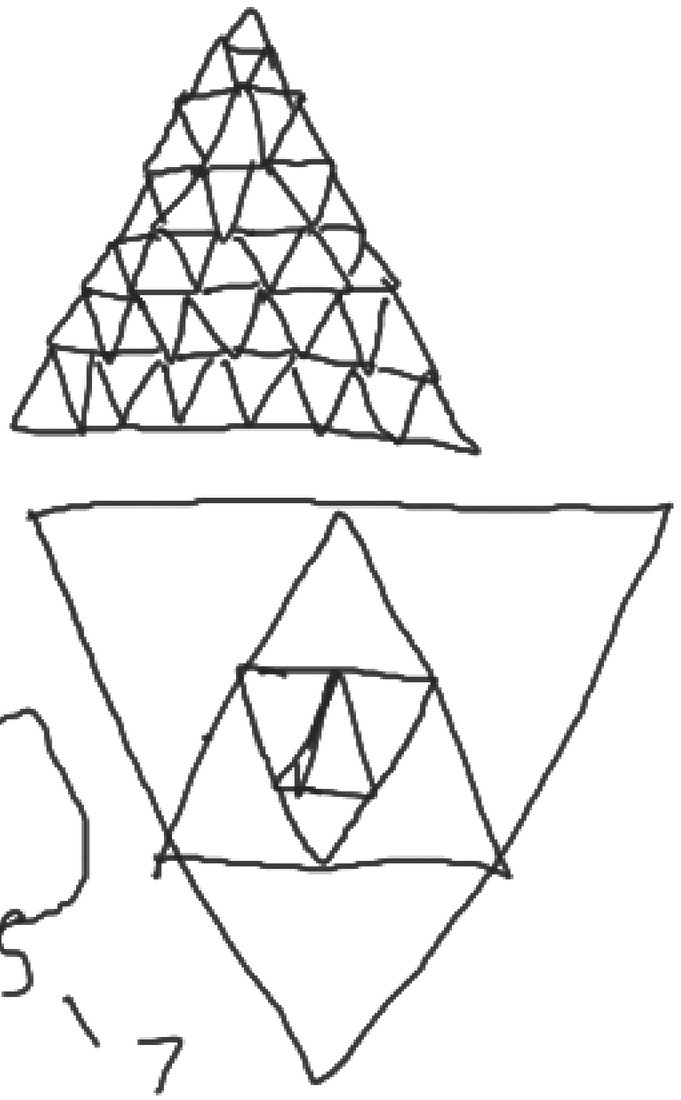
1) O(n) 2) O(n) 3) O(logn)

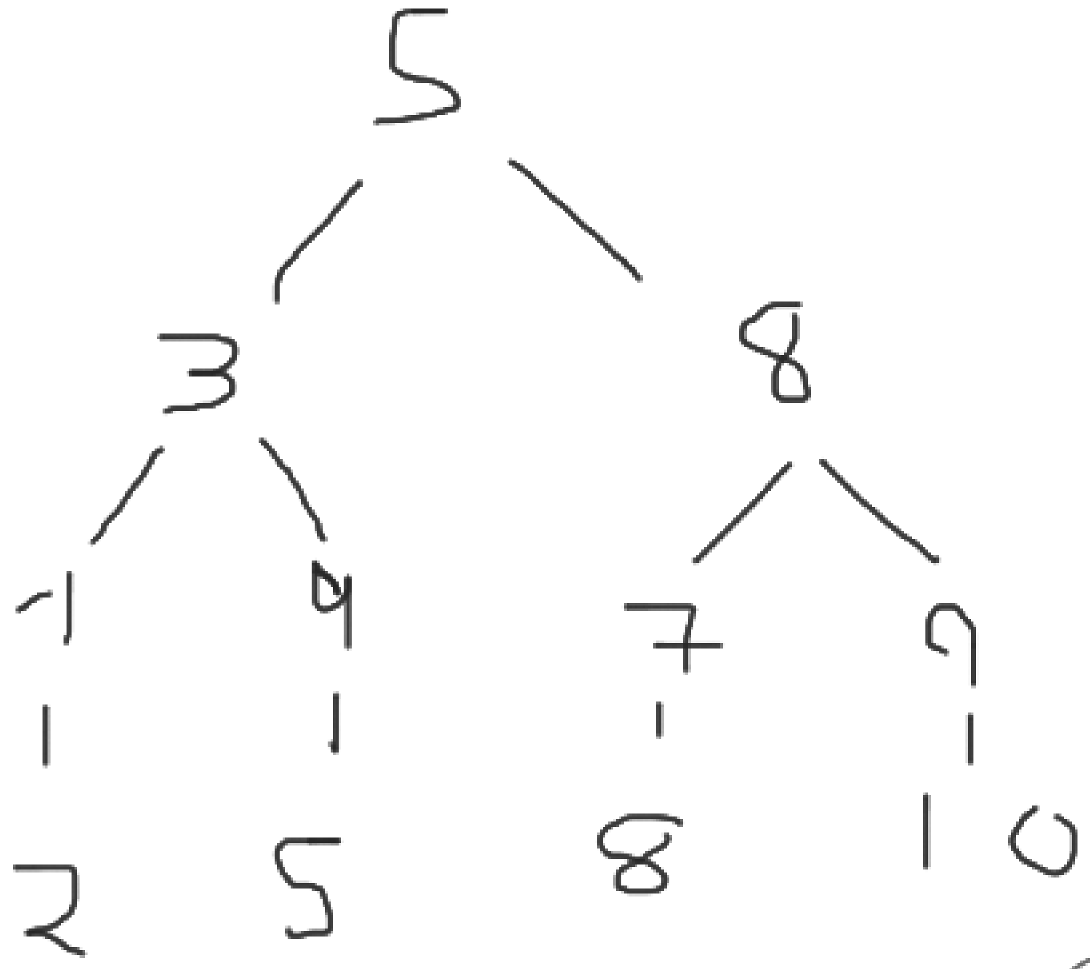




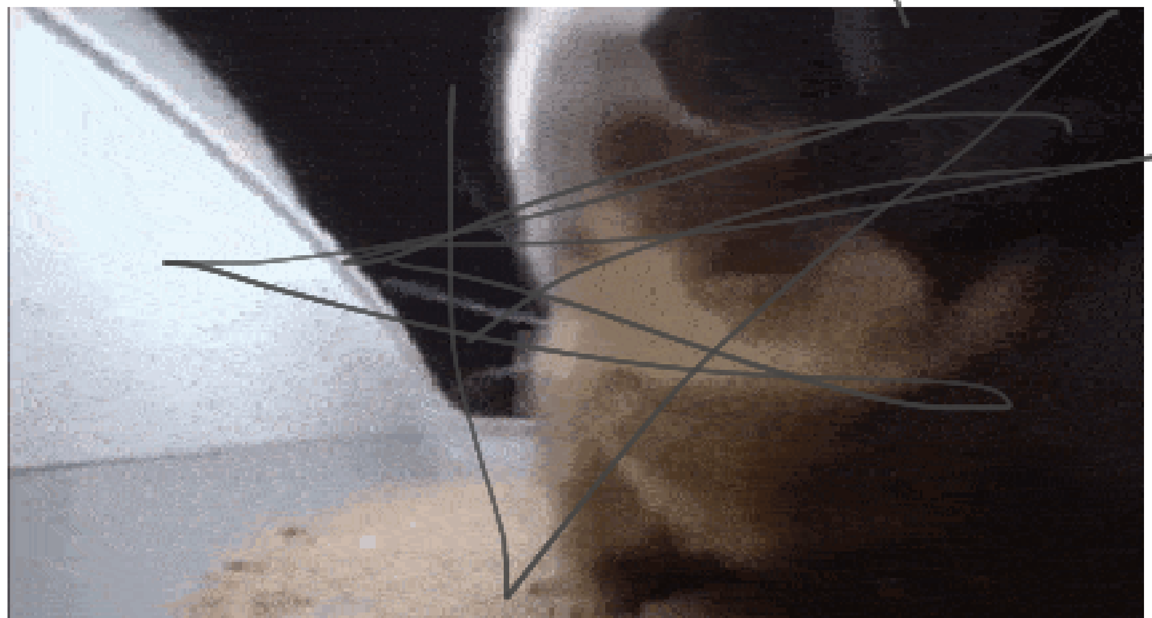


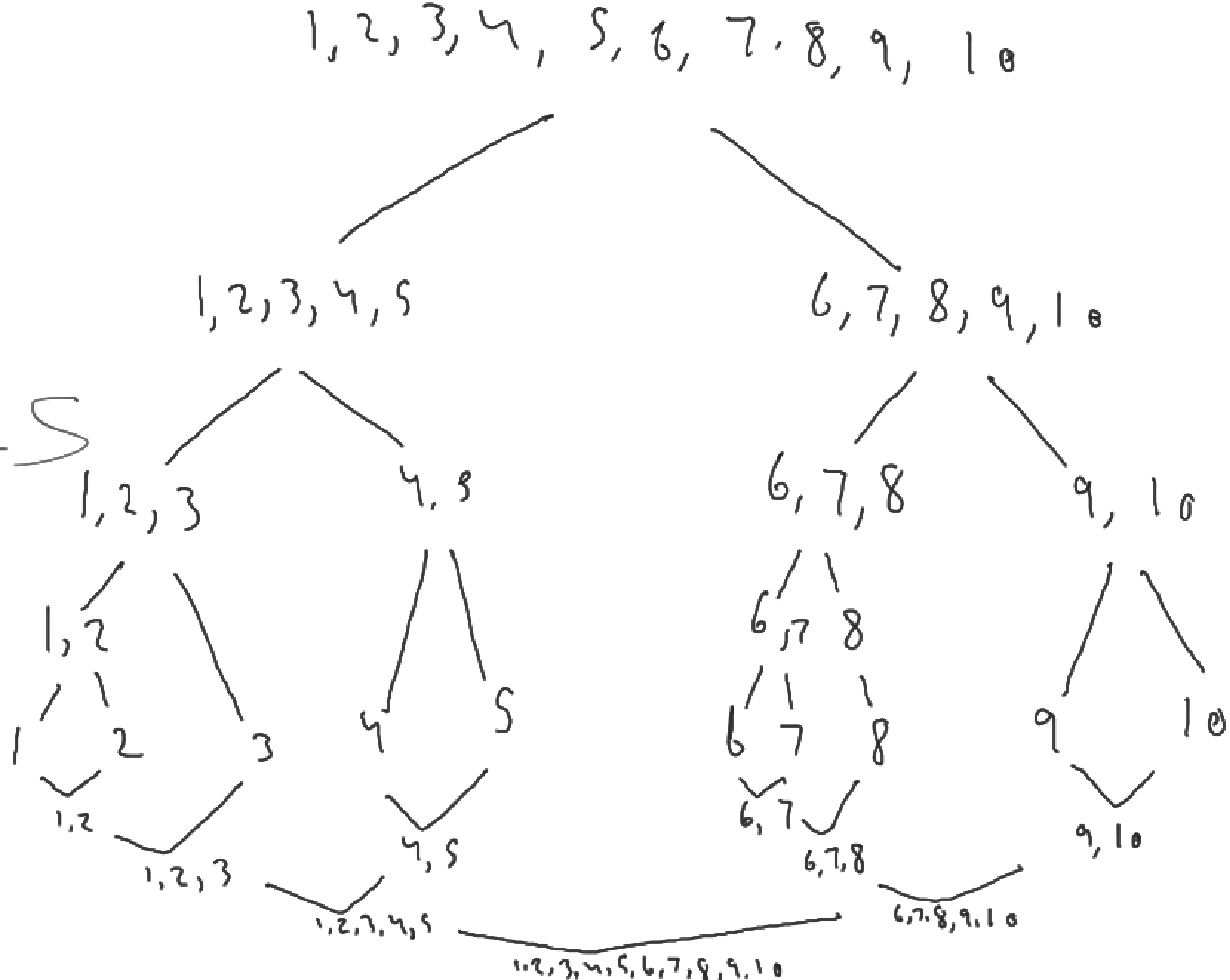


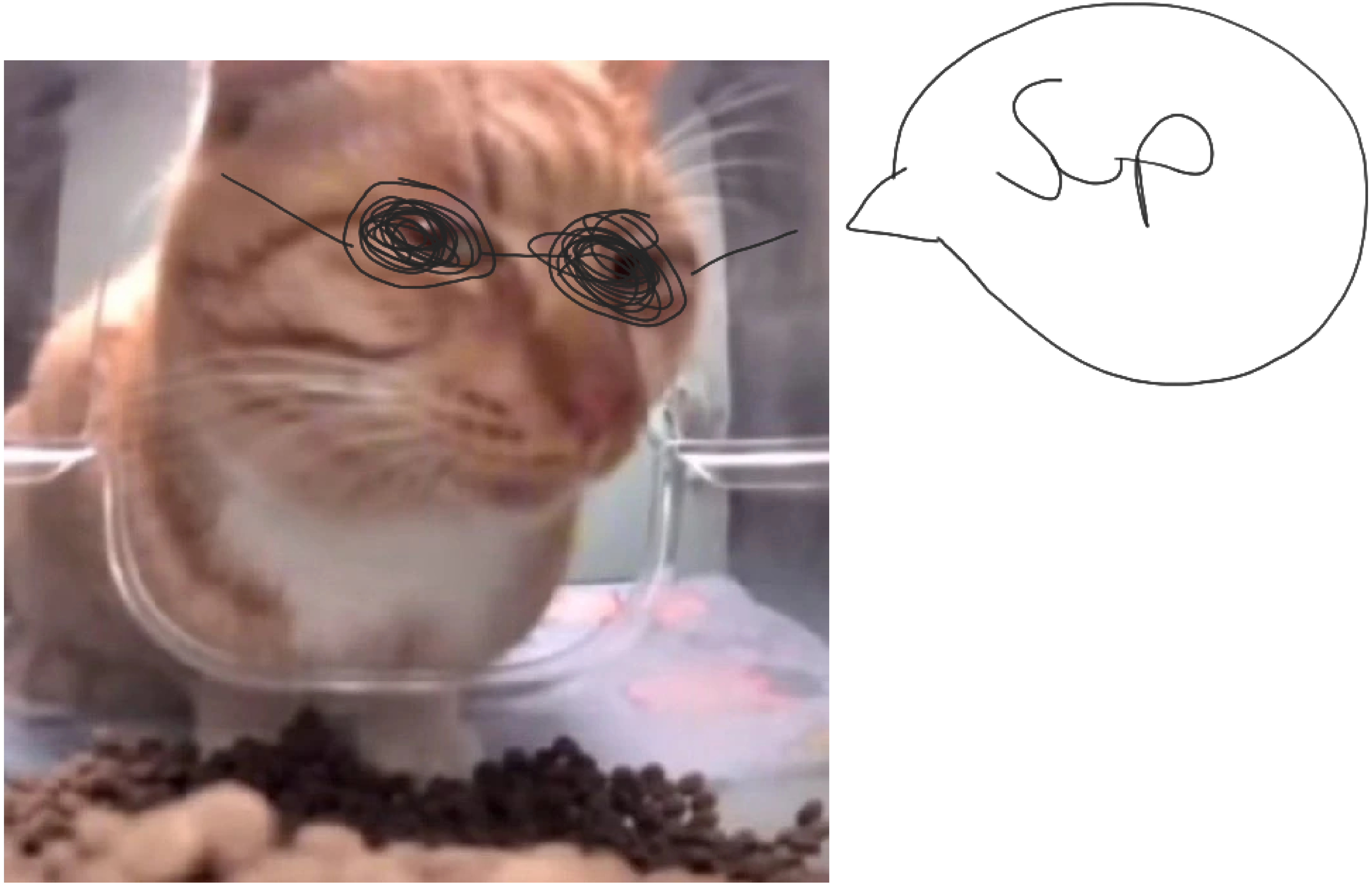



5
3
8
1
4
7
9
2
5
8
10

CS




1, 2, 3, 4, 5, 6, 7, 8, 9, 10
1, 2, 3, 4, 5
6, 7, 8, 9, 10
1, 2, 3
4, 5
6, 7, 8
9, 10
1, 2
3
4
5
6, 7
8
9
10
1
2
6
7
8
1, 2
4, 5
6, 7
9, 10
1, 2, 3
6, 7, 8
1, 2, 3, 4, 5
6, 7, 8, 9, 10
1, 2, 3, 4, 5, 6, 7, 8, 9, 10


Sup