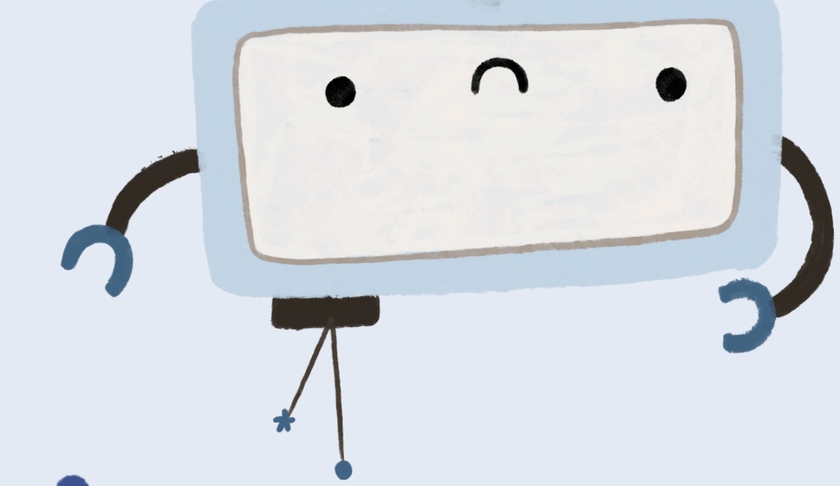
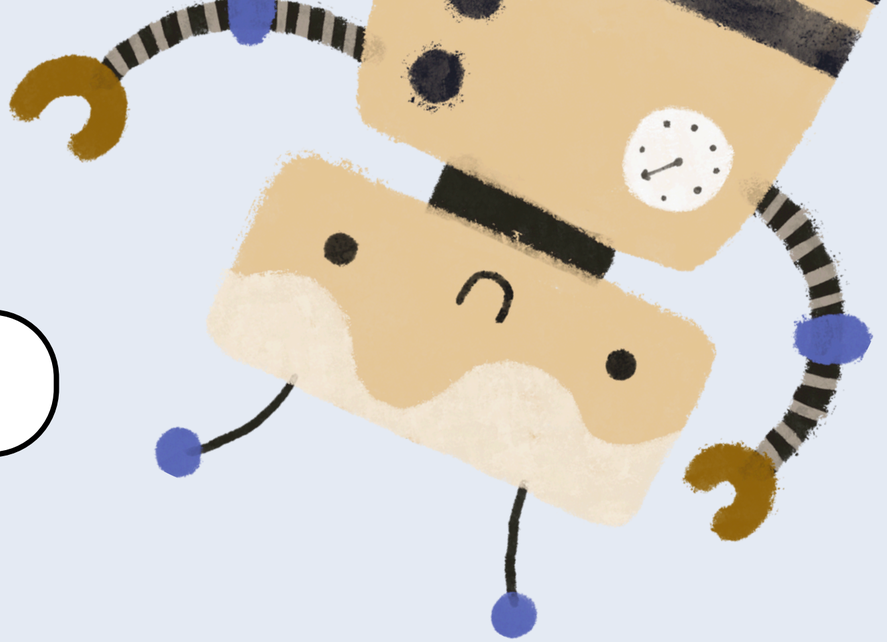
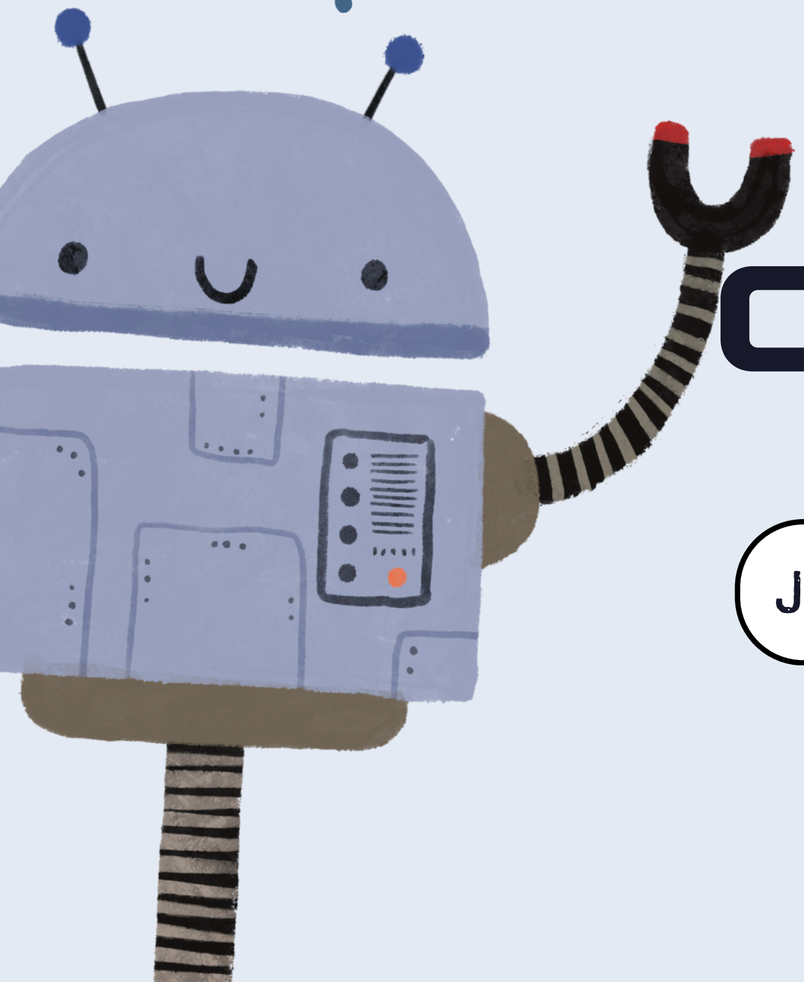


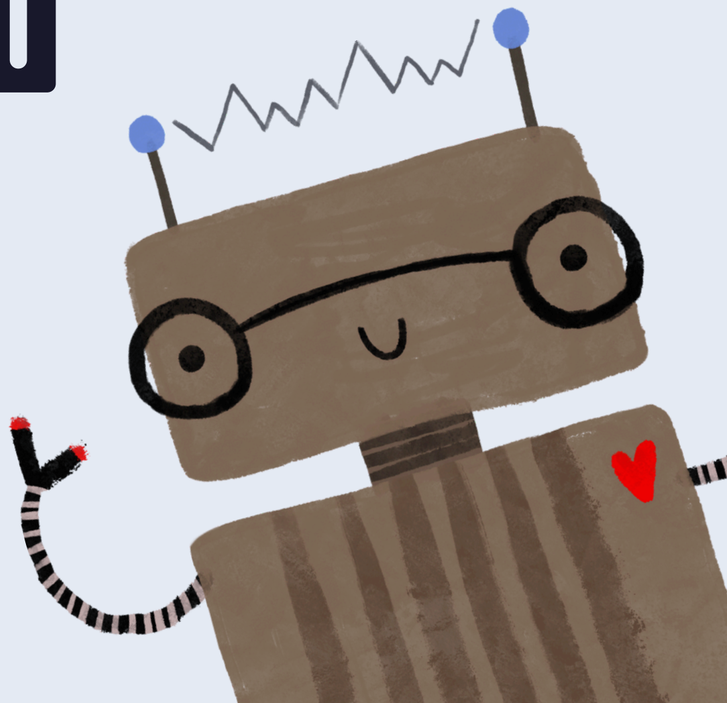


GROUP 3

SORTING COMPETETION



JOHNETTE NAGBE & FIONA HUSSEN



SCORES + TIME

ROUND 1

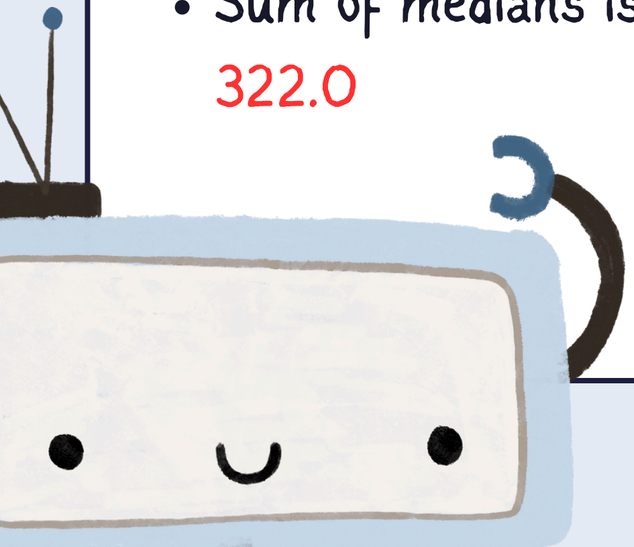
- Place 1
- Sum of places is 2
- Sum of medians is 322.0

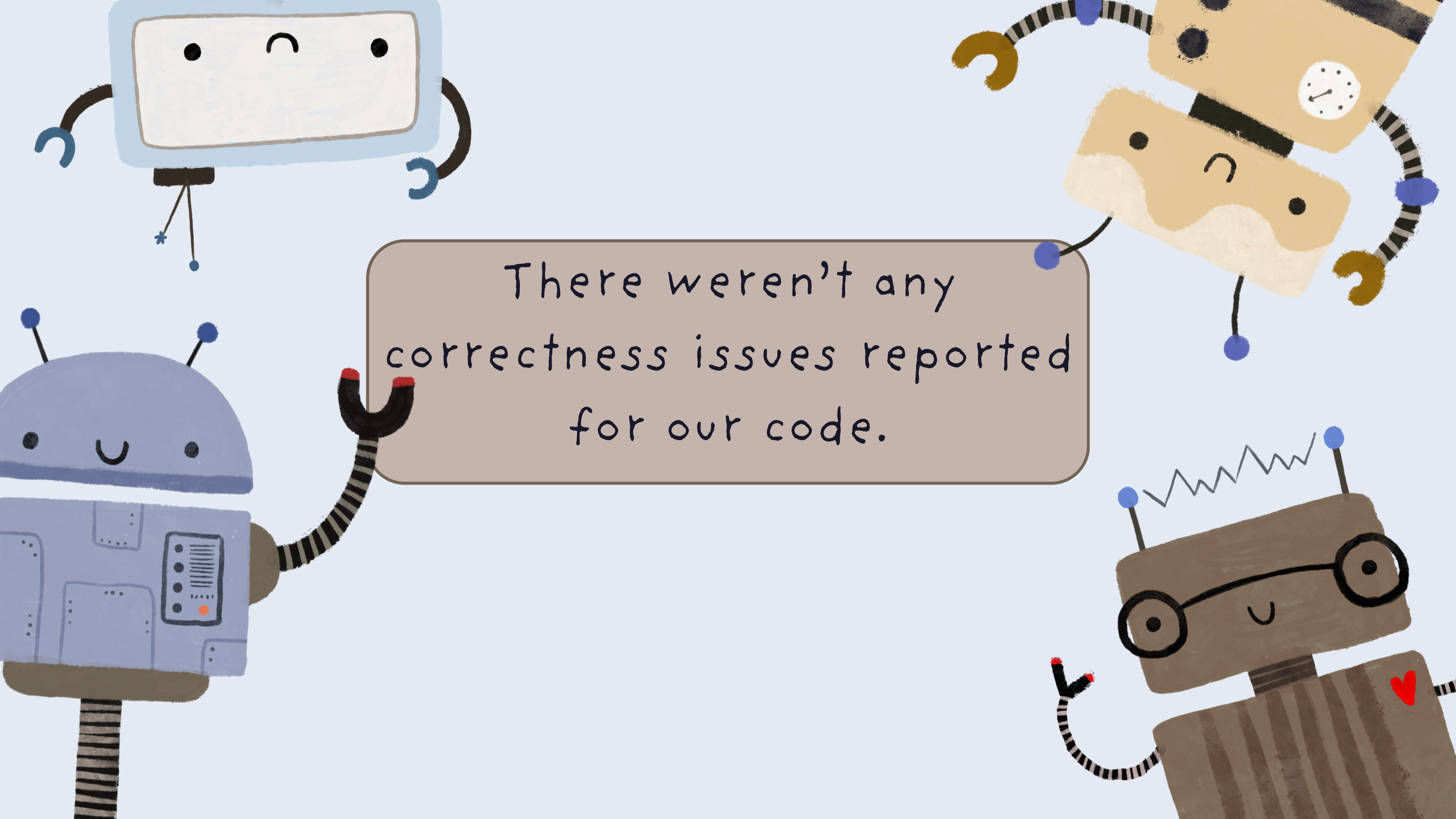
ROUND 2

- Place 1
- Sum of places is 2
- Sum of medians is 407.0

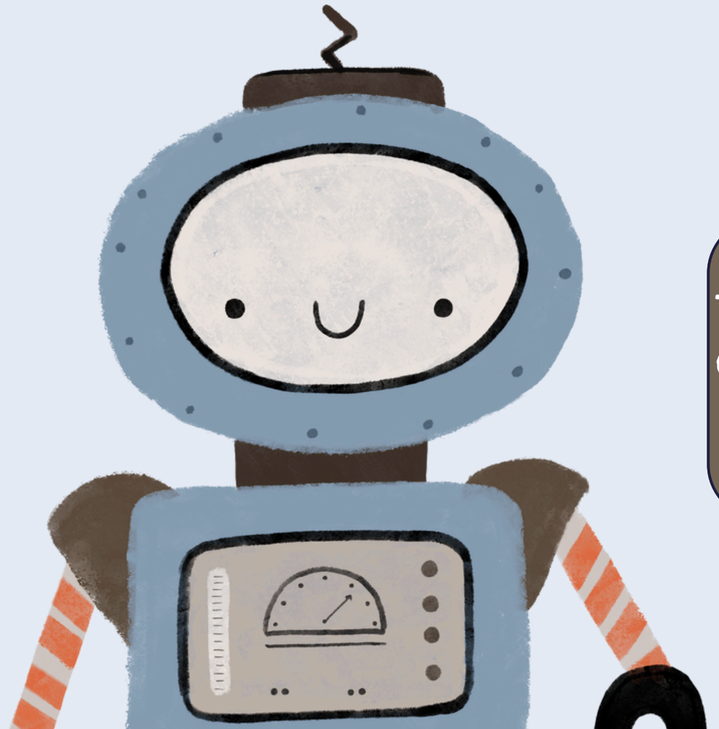
FINAL ROUND

- Place 2
- Sum of places is 5
- Sum of medians is 338.0




There weren't any
correctness issues reported
for our code.

THE ALGORITHM



We sort binary strings by:

1. Hamming distance from a target string (number of differing bits).
2. Numeric value of the string (used as a tie-breaker). split into two long values (hi and lo) for fast comparison.

This is faster and more memory-efficient than Group0's full-array sort.
directly.

3. Instead of sorting the entire array directly. Group3:

- Reassembles the sorted strings from all buckets.
- Wraps each Binary string in a object.
- Groups strings into buckets based on their Hamming distance.
- Sorts each bucket by numeric value.

TABLE IS GENERATED BY NOTION AI

Change	Location	Description
New class `BinaryString`	Added inside `Group3.java`	Stores each string's value, Hamming distance, and numeric representation (`hi` , `lo`)
Removed `SortingCompetitionComparator`	Deleted from `Group0.java`	Replaced with bucket-based sorting logic
Rewrote `sort()` method	Replaced in `Group3.java`	Uses bucket sort instead of `Arrays.sort()` with comparator
No changes to `main()` or `writeOutResult()`	Left unchanged	Sorting still happens in-place, output format is the same

GROUP 3

TIM COMPLEXITY

Let:

- = number of binary strings
- = length of each string

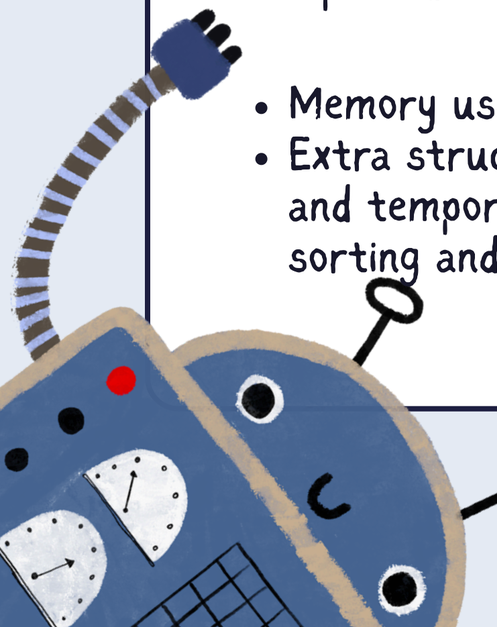
Total worst-case time:

O(n x L + nlog n)

Expected time:

O(n x L + nlog n)

- Memory usage: Linear in n, plus L+1 buckets
- Extra structures: BinaryString[], ArrayList[], and temporary long[] — all used for efficient sorting and numeric comparison



Expected-Case Time Complexity

If Hamming distances are well-distributed:

- Most buckets are small
- Sorting within each bucket is faster

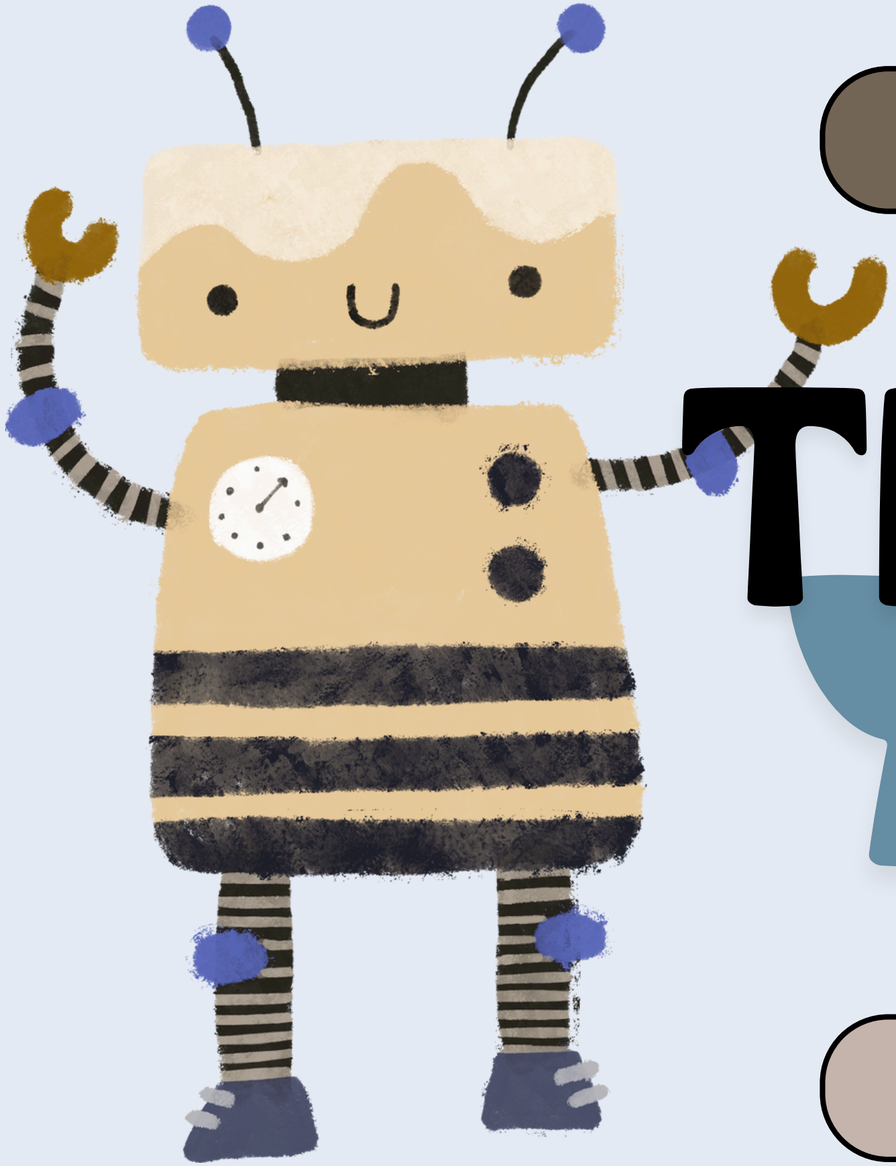
COPILOTAI

Data Sorted:

Structure	Type	Purpose	Size
`BinaryString[] wrapped`	Array	Stores each string's value, Hamming distance, and numeric value (`hi`, `lo`)	$O(n)$
`ArrayList<BinaryString>[`	Array of lists	Groups strings by Hamming distance (0 to L)	$O(n+$
`long[] bits`	Array of size 2	Temporarily holds parsed binary values	Consta nt

Worst case:

Step	Operation	Time
1. Wrap strings	Compute Hamming distance and parse bits	$O(n \cdot L)$
2. Bucket insertion	Place each string into one of $L+1$ buckets	$O(n)$
3. Sort each bucket	Sort by numeric value using `hi` and `lo`	$O(n \log n)$ in worst case (if all strings fall into one bucket)
4. Reassemble output	Flatten sorted buckets into final array	$O(n)$



Johnette Nagbe

THANK YOU

Fiona Hussen