CMSC 27200 - Guerilla Section 4

Topics

1. Some thoughts on the midterm.
2. The werewolf question.

Regarding the midterm

So I wanna talk about the midterm, and I want to organize my thoughts in three parts.

1) Observations about the midterm.
2) Pragmatics.
3) Bigger picture / overall vibes.

⟹ Observations.

⟶ How did people do?

↳ The exam was written w/ about 60 pts. as the target median

⟶ As of the last check

median 60.5 pts. bulk of distr.

mean: 62 ish pts. ⟶ Skews > median.

↳ People did better than I expected

→ What did people understand.

- People knew how to use Masters theorem to analyze recurrence relations.
- ... all the types of DFS edges.
- How to run Dijkstra's algorithm.
- What the FFT matrix is.

↳ And generally people did well on the short answer section.

→ What was hard?

- The DAG of strongly connected components
- Analyzing strange recurrences. (where the master thm doesn't quite fit).
- Algorithm design questions.

↳ overall especially

→ What does that tell me as an educator?

All the pieces separately are sticking, but there is still difficulty trying to aggregate the ideas together.

For example, people know.

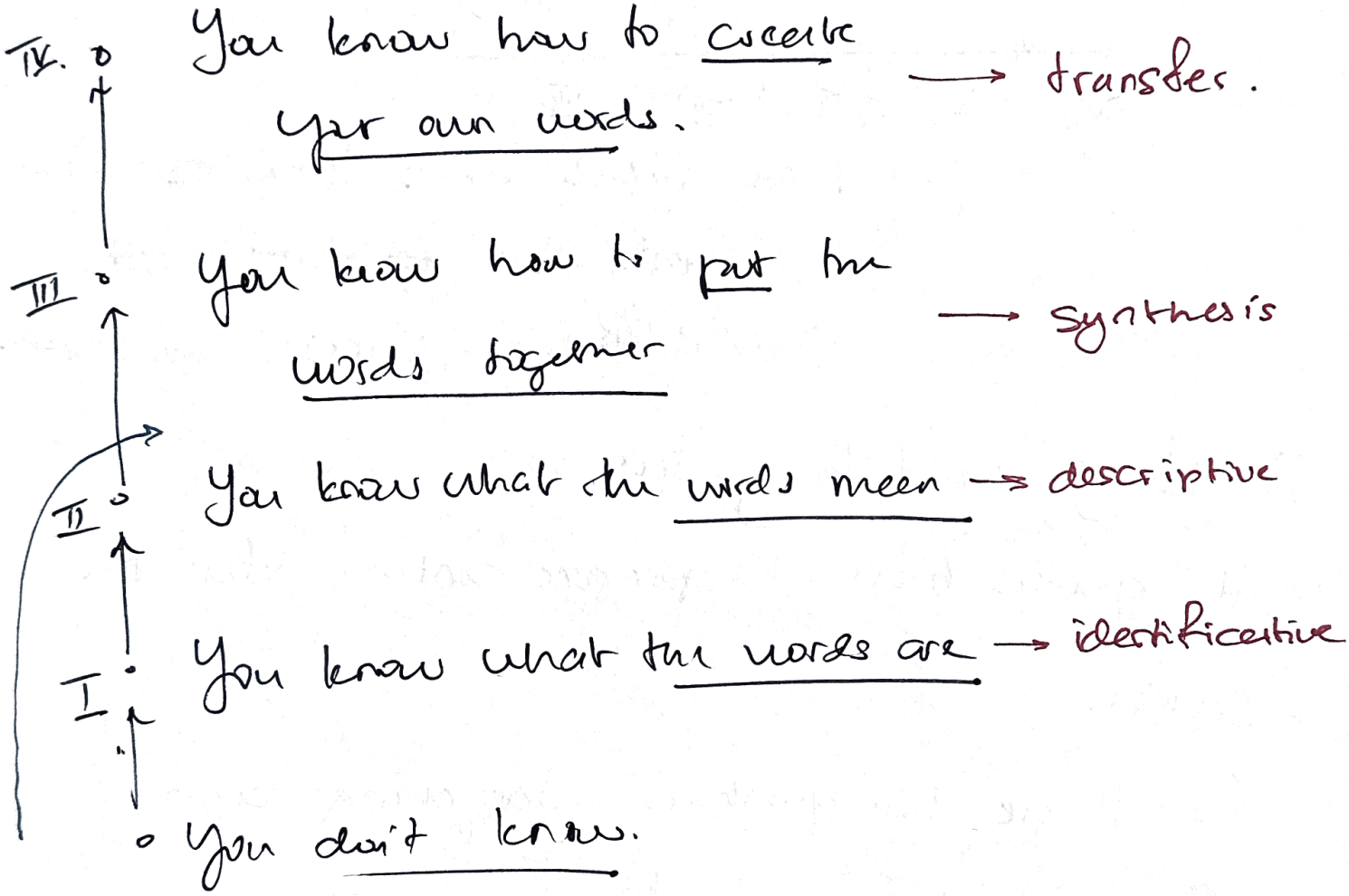
- What an FFT is.
- What dijkstra's algorithm does
- What a strongly connected component is...

But aggregating ideas and transferring them to other domains is still difficulty.

- What's the principle behind divide and conquer? Identify a subproblem that preserved well-defined structure in the input as it makes recursive calls.

→ Some of the theory behind this.

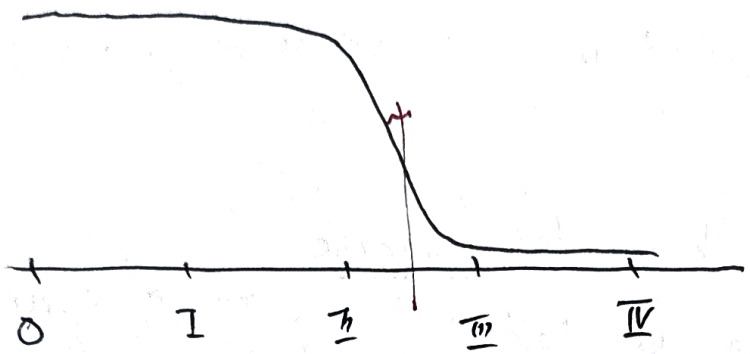
There's a certain hierarchy of understanding typically discussed in education theory.



"If you know what the words mean that's like 60% of the battle" → actually that played out precisely.

(86)

Where do student stand.



most students ~~under~~ know ~~that~~ the words but ~~not how to put~~ have difficulty putting them together

→ Why do I bring this up?

① It guides how I prepare content for the course.

↳ More new questions targeting design | ways of putting the ideas together.

② ~~↳ Guerilla seek~~ Discussion section more Summative maybe.

→ It also guides how I study.

Doing a PhD means at a certain point you learn

Any self-study things.

I) Draw up a vocab list, collect words and group them under topics.

II) Define those words (defn sheets).

↳ Describe processes / algorithms

↳ Collect and do "run this algo type questions" ~~short answer~~

III) Relate defns together

↳ "FFT is a divide and conquer algorithm because ..."

↳ Dijkstra's finds the shortest path using greedy because ...

↳ short answer type questions.

IV) Any and all practice questions (particularly from finals / midterms

↳ Algo design questions especially.

→ Try studying this way...

→ So what's my final takeaway?

↳ Are the bars alright?

↳ yeah absolutely!

→ But I don't feel alright!

↳ yeah absolutely...

⟹ Pragmatics.

Okay am I doomed? Absolutely not...

→ Is the midterm score going to be curved?

No single score in this class is curved

→ How will this affect my grade?

Important thing to emphasize.

- Course staff is absolutely not blind from the effort that you the students put in.
  - ↳ From day 1 we've said the course is fast paced and difficult.
  - → The fact that I still see you here holding on, asking questions, going to OH, doing hw, going to guerilla section, and generally working hard.

    is absolutely not lost on me.

    And demands the utmost respect.
- So when we go to design the final grade we in fact have a lot of different pointers we try.

- One policy is clustering the final. That means we reweigh the final grade to be much higher than the midterm.
  - ↳ If you perform significantly better you can conceivably walk away w/ a very good grade.
- We adjust the wt of hws relative to # of resubmissions.
  - ↳ You can also dowell if you show like striking and consistent improvement on the HWs.

→ We try to make the grading process as holistic as possible.

⟹ So what's the vibe?

→ The test was pretty hard.

↳ So if you didn't meet your expectation.

that's okay.

→ There are ample opportunities to still do well.

lower.

↳ So don't temper your expectations!

→ I can see the effort you all put in and I guarantee you it'll pay off.

↳ So don't give up.

→ If there's material / content that we can provide to better help your preparation.

↳ Let us know

→ Finally... I wanna talk about a vibe that I know will work...

↳ At the end of the exam, I saw a large

- Group of students walking out together.

Some of yall were saying tho you know I'll be at every office hours after this.

- But others were also saying alright we're gonna run these HWs together from now on.

↳ I wanna say do that & keep showing up and ~~show up~~ keep showing up together

↳ Maybe some of these study streaks will help see if yall can help eachother go through them.

- Here's an idea: Someone create a group owned doc., share it with the entire class.

↳ Collect the words and what they mean.

↳ write down how to tackle problems.
(& don't share how solve...).

↳ Post it on ed and I can help contribute to it.

And at the end of the quarter let's see how far we can get.

→ questions?

Werewolf.

Let's go over the werewolf question.

→ This was a hard algos question so let's talk about the thought process going through this.

→ So what is the process?

(94)

1. Understand the input / output.
2. Choose a design paradigm.
3. Try to fit a pattern.
   ↳ If that doesn't work fall back to first principles.

(1) Understanding the input / output.

The problem was given like this.

Input: n people each labeled human or werewolf w/ $\geq \frac{n}{2}$ humans.

Output: Identify all ~~humans~~ werewolves in as few queries as possible.

Now a query is defined like so.

It satisfies the following.

Input: A pair of TAs. $(i,j) \in [n] \times [n]$.

Output: you ask the pair (i,j) whether the other is a werewolf.

- humans will always tell the truth
- werewolves may or may not lie.

When you see something like this... it is helpful to get a feel of all the different ways a query could give you information.

⟶ Run it in cases.

could these both be human?

i responds	j responds	conclusion
j → werewolf	i → werewolf.	at least one is a werewolf.
j → werewolf	i → human.	at least one is a werewolf.
j → human	j → werewolf.	———
j → human	i → human..	both are either werewolves or human.

Conclusion: If they both claim other is human either they're both humans or they're both lying.

→ In all other cases... at least one is a werewolf.

→ Trying different inputs allow you to make helpful observations about the problem.

Another observation: humans always have start here to tell the truth.

↳ If we can find one human, then we can use that human to test each person.

→ This reduces the task down to finding one human instead of all werewolves.

→ General principle: seek observations that will simplify the problem.

We now have a well-defined algorithmic task...

Input: Given n people w/ $\geq \frac{n}{2}$ humans and rest werewolves.

Output: find one human in as few queries as possible.

→ Questions?

[2] Choose a paradigm.

Now for this task let's zoom on algorithmic paradigm. There aren't too many that we've seen up until this point.

1. Divide and conquer.
2. Graph traversal.

→ modeling the information you gain as a graph seems tricky, let's try divide and

Conquer. first.

→ First thing to try is to see if you've can fit a pattern of an algorithm you've seen prior to this problem.

→ One of the first divide and conquer algos we see is merge sort.

~> What's the rough idea?

To sort a list...

Observations:
1) divide things by 2. → half.
2) always make sure list is sorted →
3) scan through every one to merge →

1) Recursively divide the list in half.

2) In the base case: you have a singleton. → The singleton is sorted.

3) In the recursive case: iterate through both sides putting each item in order. → return a sorted list.

There are three observations about the structure of mergesort ... maybe we can design an alg. that mimics it.

* Instead of always guaranteeing list is sorted $\Longrightarrow$ Alg always outputs a human.

The next thing we need to make sure is that. when we recursively call things ... the structure of the input is preserved

↳ The only structure we have is. $\geq \frac{n}{2}$

$\longrightarrow$ This could be a inductive hypothesis.

If the input of TAs has $> \frac{n}{2}$ humans then alg will output a human.

Let's design an algorithm like that ...

1) base case: Suppose we have n=1 TA.
if $> \frac{n}{2}$ TAs are human. then that TA is human ⟶ output that TA.

2) inductive case:

mergesort suggests ⟶ by splitting the TAs into two groups. and running algo recursively.

⟶ Suppose we do that. Say

$S = \{1 \dots n\}$

is our list of TAs and we run algo on

$S_1 = \{1 \dots \lfloor n/2 \rfloor\}$ $S_2 = \{\lfloor n/2 \rfloor + 1 \dots n\}$.

~~If we run our hypothetical algo on it~~
~~then by the inductive hypothesis~~

~~Algo(S_1) ⟶ h_1.~~
~~Algo(S_2) ⟶ h_2.~~

Alg$(S_1) \to i_1$ Alg$(S_2) \to i_2$.

Can we run an Inductive hypothesis on the recursive calls to S_1 and S_2?:

↳ Precisely: can we guarantee that in S_1, more than half are humans? and in S_2 more than half are humans?

↳ because if so then by inductive hypothesis ... we're done both are human and we can output anyone...

→ Answer is no! you might not be able to guarantee both lists have $> \frac{n}{2}$ humans in it.

Imagine if you have exactly $\lfloor \frac{n}{2} \rfloor + 1$ humans for n odd. At least one list will have

$$\leq \frac{n}{2} \text{ humans.}$$

Does that mean we're doomed? No! because mergesort suggests for us to keep trying.

↳ Remember there's a non-trivial merge step.

→ Also -- even though we can't guarantee both lists will have $> \frac{n}{2}$ human TAs.

↳ at least one will

↳ by the IH at least one list will ~~have~~ return a human.

→ we just need our merge step to identify the human!

→ questions?

How do we identify which candidate TA is a human?

→ remember in the list $S = \{1 \ldots n\}$ you're given. a majority are humans.

↳ And humans always have to tell the truth.

→ For each candidate i_1, i_2 ~~test~~ query them w/ each $j \in S$. i.e. run

query (i_1, j), query (i_2, j) $\forall j \in S$.

↳ If i_1 or i_2 is a human, a majority of queries will return ~~human~~. i_1 | i_2 is human.

→ Output the candidate that gets a majority of queries to vote human.

→ Questions?

(a4)

Now we can write the pseudocode

INPUT: A list of IAs $S = \{1 \dots n\}$.

DO: the following

1. Split the list into

$$S_1 = \{1 \dots \lfloor n/2 \rfloor\}$$

$$S_2 = \{\lfloor n/2 \rfloor + 1 \dots n\}$$

2. Recursively call FINDHUMAN(S_1), and FINDHUMAN(S_2) to get candidates i_1, i_2.

~~3. let k_1, k_2 be~~

~~k_1 = # of queries Query(i_1, j)~~

3. Initialize counters $k_1, k_2 = 0$.

4. For each $j \in S$.

- If Query(i_1, j) outputs i_1 human, increment $k_1 += 1$.
- If Query(i_2, j) outputs i_2 human

increment $k_2 += 1$.

5. If $k_1 > \frac{n}{2}$ output i_1, otherwise output i_2.

→ What are the principles?

Notice how we kinda just gave a proof sketch that this works.

#1) The proof forms before the algorithm.

In writing our sketch, we chose a firm well defined inductive hypothesis.

#2) A divide and conquer alg gets designed around an inductive hypothesis.

To get the precise shape of the algo. → compared it to mergesort.

#3) Use divide and conquer algorithms you know well to guide the design of your new algo.

Finally the rest area is trying to answer questions "and make it make sense"

↳ The recursive step is implemented to make sure that the inductive hypothesis holds

→ questions?

⇒ Proof sketch.

Let's summarize our proof sketch...

Claim: (for any n) If alg receives S s.t. $> \frac{n}{2}$ TAs (out of n TAs) are human then alg outputs a human.

Pf:

base case: for $n=1$, if $> \frac{1}{2}$ TAs are human

then that one TA is human.

Inductive Step: strong ind. suppose $\forall n' < n$, claim holds.

- If S has $> \frac{n}{2}$ humans. then by pigeonhole principle. at least one of S_1, S_2 will have $> \frac{|S_1|}{2}$ human TAs or $\frac{|S_2|}{2}$ human TAs.
- WLOG let it be S_1. By IH, alg(S_1) will output i, a human TA.
- Because $> \frac{n}{2}$ humans in S, ~~the # of queries~~ $k_1 > \frac{n}{2}$. Since each → ~~Alg outputs i.~~ human has to tell the truth.

→ Alg outputs i. Ω.

⟹ Running time analysis.

Let's now compute how many queries we need...

$$T(n) = 2 \cdot T(\lceil n/2 \rceil) + 2n.$$

- 2: 2 recursive calls
- $\lceil n/2 \rceil$: two lists of $\leq \lceil n/2 \rceil$ lists.
- $2n$: $2n$ total # of queries in a single

This looks like a job for the master thm.

$a = 2, \; b = 2 \quad d = 1$

$\log_b a = \log_2 2 = 1 = d.$

⟶ $O(n^d \cdot \log n) = O(n \log n)$ queries.

⟶ Questions?

Okay but what if I don't remember mergesort?

→ Let's go through an algorithm that can be derived from first principles.

What's the mantra for divide and conquer.

→ reduce the problem size while preserving ~~its structure.~~ the input's structure so that the problem can be solved recursively

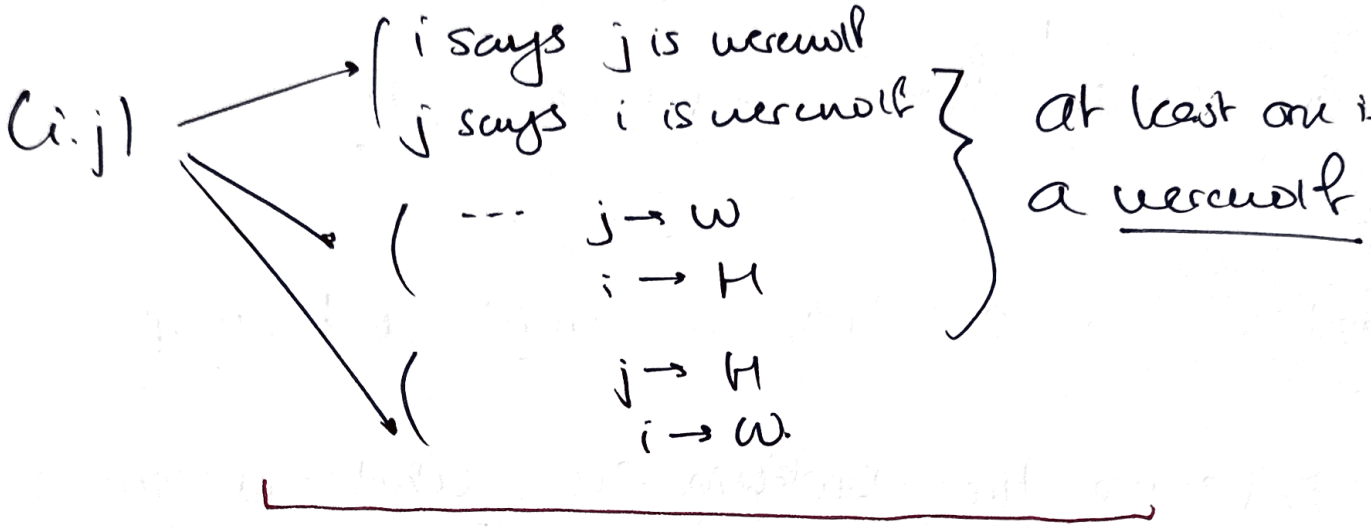
→ ~~In our case we want to reduce the # of TAs we have to test~~

Let's go back to the task of finding 1 human.

→ We want to reduce the # of TAs in our list while still maintaining > half are human.

(10)

Imagine running a single query, and recall what we deduced in our case by case analysis.



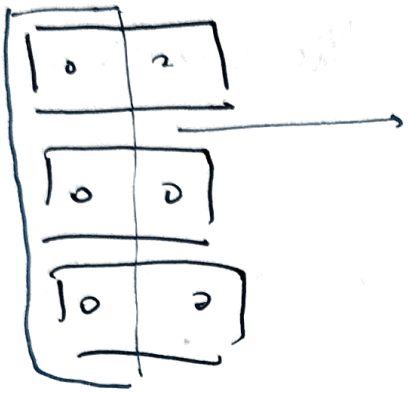
Ⓔ When you get a query result like this imagine throwing (i, j) from your list.

↳ what happens to the # humans relative to werewolf?

⟶ $> \frac{n}{2}$ humans because you threw out at least one werewolf!

- Best case you threw away two werewolves
- Worst case you threw away an equal # of humans and werewolves

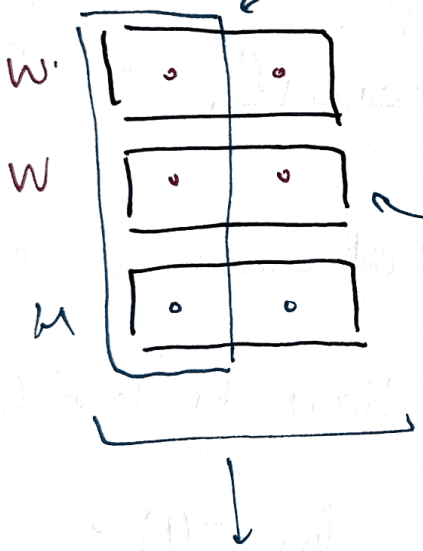
(112)



claim: for the set of people you kept, the majority are ~~not~~ human

How do you know this?

→ Suppose not!



then in this set majority were werewolves!

but because each pair reported both human.

then you know both ~~of~~ in the pair are werewolves!

that means out of this entire set, majority werewolves.

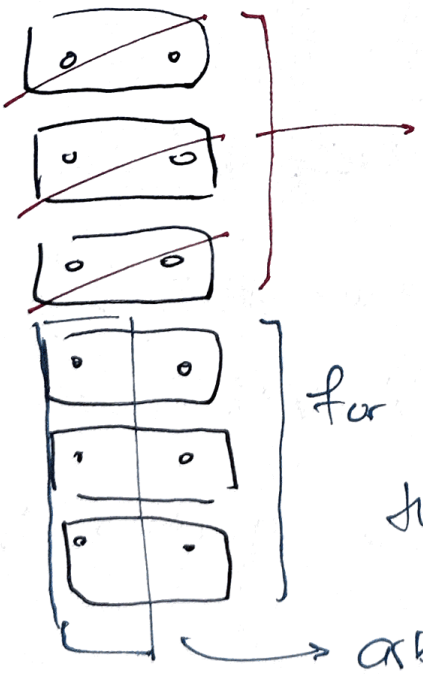
→ Contradicting fact that by tossing away all pairs who didn't say both Human, you preserve majority human.

→ So then what happens if they both report the other is human?

- Then either both are werewolves or both are humans.
- Arbitrarily keep one → because ~~if one~~ if you know one is human, you know both ~~case you keep one human, in the other you throw away a werewolf~~ are human.

→ Let's try to use this observation to reduce the # of TAs we have to test by half

↳ Imagine pairing up all the TAs.



for all who return ~~at least~~ not both human. Throw them away.

for the remaining we know the majority are human.

↳ arbitrarily pick one half.

By doing this we've reduced our problem size by at least half!

→ what if n is odd?

↳ you can always reduce to n is even by using O(n) additional queries.

- Take the odd TA i^{th} and test them w/ every other TA. j.
- If majority say i^{th} is human.
  - ↳ Then i^{th} is human because you know majority of TAs won't lie.
- If majority say i^{th} is accwolf
  - ↳ Toss them away → because maj. of TAs won't lie.

(114)

We now have our algorithm.

PROC: Find human.

INPUT: ~~bbb~~ TAs. $S = \{1 \ldots n\}$.

Do: the following.

1. If ~~$|S|=0$~~, ~~beg a o~~ $|S|=1$, output that one TA.
2. If $|S|$ is odd. Let i^* be the first TA.
   - Run a query (i^*, j) $\forall j \in S$ w $j \neq i^*$.
   - If $> \frac{|S|}{2}$ of queries say i^* human
     return i^*
   - Else remove i^* from S.
3. ~~Pair the TAs together and run~~
   ~~query $(i, i+1)$~~
   ~~for $i = 1, 3, 5 \ldots n-1$~~.
3. Initialize $S' = \emptyset$

4. for $i = 1, 3, 5, \ldots, n-1$: do

   - Query $(i, i+1)$
   - If both output true other is human

     add $i \rightarrow S$.

   - Else ignore.

5. Return FINDHUMAN (S').

$\Longrightarrow$ Running time analysis.

Notice on each recursive call, we require at most $2n$ queries (really $n + \frac{n}{2}$)., to reduce the problem size by half. Thus we make one recursive call.

$$T(n) = T(\lceil n/2 \rceil) + 2n.$$

$\longrightarrow$ $a = 1, b = 2, d = 1$.

$\log_2(1) = 0 < d \longrightarrow O(n)$ queries.