INTRODUCTION TO ALGORITHMS AND THEORY OF COMPUTING

Credit: SMBC Comics by Zach Weinersmith

THE FAMOUS PERSON PROBLEM

Person p is **famous** iff everyone in this class knows p, but p knows no one else in this class.

A single **query** consists of taking a pair of people ⟨p, q⟩ from the class, asking if p knows q, and receiving a response.

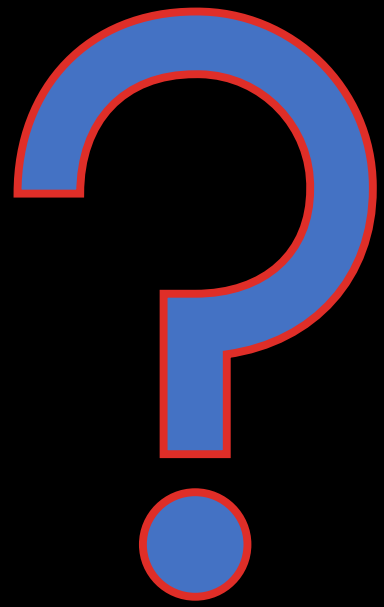
You want to determine all the famous people in a class of size n, using the minimum possible number of queries.

Who do you suppose is famous in this class?

- Probably me

How would we test this hypothesis?

- For every person in the class, ask if they know me, and if I know them. If any queries fail, I'm not famous. Otherwise, I am.



A FIRST ATTEMPT

Turns out I do know quite a few of you, so I'm not famous.

Does that mean nobody is famous?

- Not necessarily. There might be a well-known student who entered the room late and is wondering who I am.

How would we alter our test to find all famous people in the class?

For all people in the class p
    For all people in the class q, where p ≠ q
        If p knows q, then p is not famous
        If q doesn't know p, then p is not famous
    If p is famous, add p to the list
Return our list

SEARCHING FOR A BETTER SOLUTION

Is this a good algorithm?

- It takes $\Theta(n^2)$ queries. We can probably do better.

How many queries does our algorithm use, exactly?

- $2n \cdot (n-1)$

How can we improve our algorithm?

- We're asking each query twice. If we are clever about how we organize our queries, we can reduce this to $n \cdot (n-1)$ (or, n permute 2) queries.

How many famous people can there be, maximum?

- One. Two would have to both know and not know each other, so we can stop the algorithm after finding our first famous person.

How else can we improve our algorithm?

A BETTER SOLUTION

If p knows q, then we know that p is not famous.

If p does not know q, is there anything we can deduce?

- q is not famous!

With a single query, we can eliminate someone!

1. Maintain a list of famous candidates, initially everyone
2. Take any pair ⟨p, q⟩ from the list, or go to step 5
3. Check if p knows q. If so, remove p. Otherwise, remove q.
4. Return to step 2
5. There is one person on the list now: c

Is c famous?

- Not necessarily.

A BETTER SOLUTION, CONT.

1. Maintain a list of famous candidates, initially everyone
2. Take any pair ⟨p, q⟩ from the list, or go to step 5
3. Check if p knows q. If so, remove p. Otherwise, remove q.
4. Return to step 2
5. There is one person on the list now: c
6. For all people q ≠ c:
7. Check if c knows q. If so, return nullptr
8. Check if q knows c. If not, return nullptr
9. Return c

How many queries does this take, in the worst case?

$3 \cdot (n-1)$

HOW TO WRITE ALGORITHMS

1. Pseudocode (such as what we wrote on the prior slides) is both acceptable and recommended. You can write in code, pseudocode, or English, as long as you provide enough detail.
2. What is "enough detail?" A reasonably competent programmer should be able to take your answer and code it up in a language of their choice.
3. Algorithms is learned with **practice**. If you think you must have a "Eureka" moment to answer an Algorithms problem on a test, that is merely a sign that you need to practice more.

STABLE MATCHING

There are n men and n women signed up on a dating site.

- Each user has ranked all of the members of the opposite sex from 1 to n.
- We need to form n male/female pairs so that everyone is in exactly one pair.

Consider the case with n=2, with men m_1 and m_2, and women w_1 and w_2.

- Suppose both women rank m_1 higher, and both men rank w_1 higher.
- If we try to match m_1 with w_2, there is something inherently wrong with our solution.
- In fact, m_1 and w_1 have no reason to go along with our proposed matching: they would dump their respective partners and match together.

THE GALE-SHAPLEY ALGORITHM

If m_j is matched with w_a, and m_k is matched with w_b, then the matching is **stable** if at least one of the following statements isfalse:

- m_j prefers w_b to w_a.
- w_a prefers m_k to m_j.

The **Gale-Shapley Algorithm** provably finds a stable matching.
While there is an unmatched male m:

1. m asks out his highest-ranked woman w whom he hasn't asked out yet.
2. If w is unmatched, she says yes.
3. If w prefers m to her current match, she says yes and breaks up with her current match.
4. Otherwise, w says no.

PROOF FOR STABILITY

Everyone will be matched:

- Women always say yes if unmatched, and men keep asking women out until someone says yes.

Assume (by way of contradiction) that the matching is unstable.

- Let the unstable pairing be (Alice, Bob) and (Charlie, Debra), where Alice and Charlie prefer each other to their current match.
- Charlie would have asked Alice out before Debra. Either Alice said no (because she was in a more desirable match), or she later broke up with him (because she was offered a more desirable match).
- Either way, Alice would be in a more desirable match (not a less-desirable one). She will only ever change partners if offered an even more desirable match. Contradiction!

MALE-OPTIMALITY

The Gale-Shapley Algorithm is **male-optimal**: out of all possible stable matchings, the Gale-Shapley algorithm finds the best one for all men simultaneously.

- It is also **woman-pessimal**: out of all possible stable matchings, it finds the worst one for all women simultaneously.
- The proof for this is in the textbook if you're interested.
- You could swap these properties if you had the women ask out the men.

If you were implementing this algorithm for your job, which version would you implement?

YOUTUBE AND SELF-DRIVING CARS

As Computer Scientists, our algorithms directly affect the lives of many people.

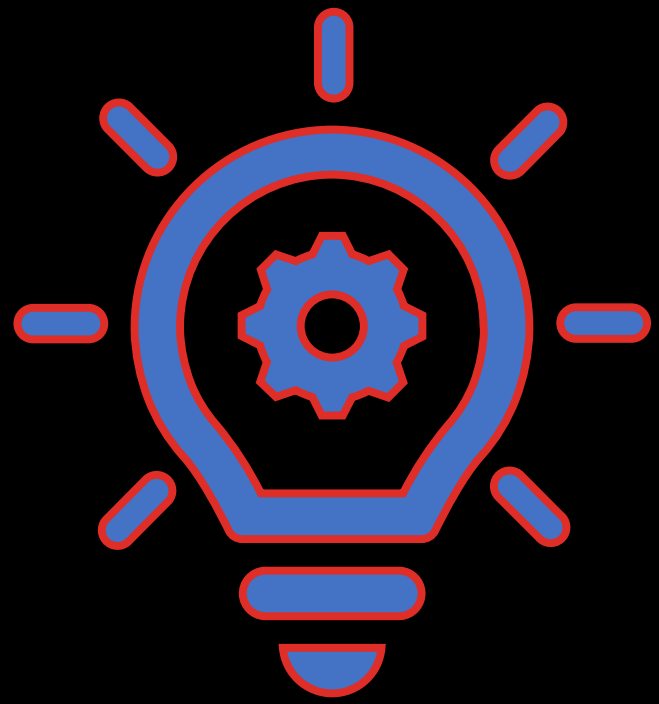
- Simple decisions about implementation can have a **profound** impact.
- Be *aware of this*, consider things carefully, and be willing to consult experts on fairness if appropriate.

Some examples of this:

- Video games are often created to produce addictive behavior
- Machine Learning algorithms, when trained on biased data, can produce biased/racist results.
- Facebook was a major platform for sharing fake news.
- The YouTube algorithm for recommending videos has radicalized people who are only ever exposed to one viewpoint.
- Self-driving cars must worry about ethical concerns such as whether to prioritize the safety of the passenger or pedestrians.

CORE COURSE QUESTIONS

1. Given a problem, how do we produce an algorithm to solve it?
2. Given a problem and algorithm, can we prove that the algorithm correctly solves the problem?
3. Given an algorithm, will it terminate in a reasonable amount of time?
4. What is a reasonable amount of time, anyway?
5. Are there problems which cannot be solved in a reasonable amount of time?
6. How could we identify such problems?
7. Are there problems that cannot be solved at all?



TAKE-HOME PRACTICE

Improve the algorithm for the Famous Person Problem to require only $3 \cdot (n-1) - \log_2 n$ queries.

Prove that the Famous Person Problem cannot be solved with less than $\Theta(n)$ queries.