CSCI 270 Lecture 1: Introduction and Ethics

Figure 1: SMBC Comics by Zach Weinersmith

Definition: Person p is **famous** iff (if and only if) everyone in this classroom knows p, but p does not know anyone else in this classroom.

Definition: A single **query** consists of taking a pair of people (p, q) in the classroom, asking if p knows q, and receiving a response.

Problem Formulation: Determine all the famous people in a classroom with n people, using the minimum possible number of queries.

1. Who in this classroom do you suppose is famous?
2. How would we test our hypothesis to the previous question?
3. If our hypothesis is wrong, does that mean nobody is famous?
4. How can we generalize our test to determine all famous people in the classroom?

A First Attempt:

```
1: For all people in the class p:
2:     For all people in the class q, where p ≠ q:
3:         Check if p knows q. If so, p is not famous.
4:         Check if q knows p. If not, p is not famous.
5:     If p is famous, add p to the list of famous people.
6: Return our list of famous people.
```

Follow-up Questions:

1. Is this a good algorithm?
2. What does it mean for an algorithm to be "good?"
3. How many queries does our algorithm use (exactly)?
4. How can we improve our algorithm?
5. How many famous people can there be, maximum?
6. If p does not know q on line 3, is there any information we can deduce?

A Better Attempt:
1: Maintain a list of famous candidates, intialized to everyone in the classroom.
2: Take any pair (p, q) from the list (if not possible, go to step **5**).
3: Check if p knows q. If so, p is not famous. If not, q is not famous.
4: Remove the non-famous person from the list, and return to step **2**.
5: There is now one famous candidate c.
6: For all people in the class q where $c \neq q$:
7:   Check if c knows q. If so, nobody is famous. Return null.
8:   Check if q knows c. If not, nobody is famous. Return null.
9: c is famous. Return c.

Followup Questions:

1. How many queries does this algorithm use in the worst case?
2. Is this better than our old algorithm?
3. Is this **always** better than our old algorithm?

Important Points

1. When writing algorithms, pseudocode is acceptable (in fact, it is recommended). You can also write in code, or in English, as long as you provide enough detail.
2. If a reasonably competent programmer could take your answer and code it up in a language of their choice, then your answer is acceptable.
3. Algorithms is learned with **practice**. If you think that 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 problems.

Stable Matching

Suppose we have n women and n men signed up on an online dating site.

- Every user has ranked all of the members of the opposite sex from 1 to n.
- The site needs to create male/female pairs so that everyone is in exactly one pair.

Suppose $n = 2$.

- Suppose both women ranked m_1 higher, and both men ranked w_1 higher.
- If we try to match m_1 with w_2 and m_2 with w_1, there is something innately wrong with our solution.
- m_1 and w_1 will dump their respective partners and match together.
- If this situation doesn't happen, we say that this is a **stable matching**.

More generally, if m_i is matched with w_a, and m_j is matched with w_b, at least one of these statements should be false:

- m_i prefers w_b to w_a.
- w_b prefers m_i to m_j.

The Gale-Shapley algorithm for finding a **stable matching**:

1. While there is an unmatched man m, repeat:
2. m asks out the first woman w on his list whom he has not yet asked.
3. if w is unmatched, she says yes.
4. if w is already matched, but prefers m, she says yes and breaks up with her current match.
5. otherwise w says no.

The Gale-Shapley algorithm is provably a **stable matching**:

- Every woman will be matched, since a woman always says yes if they are unmatched, and men will eventually ask everyone out until someone says yes.
- Assume there is an unstable pairing $\langle$ Alice, Bob $\rangle$ and $\langle$ Charlie, Debra $\rangle$, where Bob and Debra prefer each other to their current matches.
- Bob would have asked out Debra before Alice, so Debra either said no, or broke up with Bob prior to this.
- Women only ever improve their match, so it is impossible that Debra prefers Bob to her current match.

The Gale-Shapley algorithm produces the **best possible** stable matching for all men simultaneously, and the **worst possible** stable matching for all women simultaneously.

- The algorithm becomes best for women and worst for men if you have women do the asking.
- If you were implementing this algorithm for your job, which version would you make?

What do YouTube and self-driving cars have in common?

- As Computer Scientists, our algorithms directly affect the lives of many people.
- Simple decisions about implementation, and where we get our data, have a profound impact.
- Be aware of this, consider things carefully, and sometimes consult experts on fairness.

What are some examples of this?

- Video games are often designed to create addictive behavior.
- Machine Learning algorithms, when trained on biased data, can produce biased/racist results.
- Facebook was a major platform for sharing fake news.

Core Course Questions

1. Given a problem, how do we produce an algorithm to solve it?
2. Given a problem and an 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 would we identify such problems?

Extra Problems

1. Improve the algorithm for the Famous Person Problem to require only $3(n-1)-\log n$ queries.
2. Informally argue why the Famous Person Problem cannot be solved in less than $\theta(n)$ queries.
3. Prove for any integer n: n is odd if and only if $3n+1$ is even.
4. Use Induction to prove that $\sum_{i=0}^{n} \frac{1}{2^i} < 2$.
5. Chapter 2, exercises 3,4,5,6.
6. Challenge problems: Chapter 2, exercise 8