CMSC 28100

Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

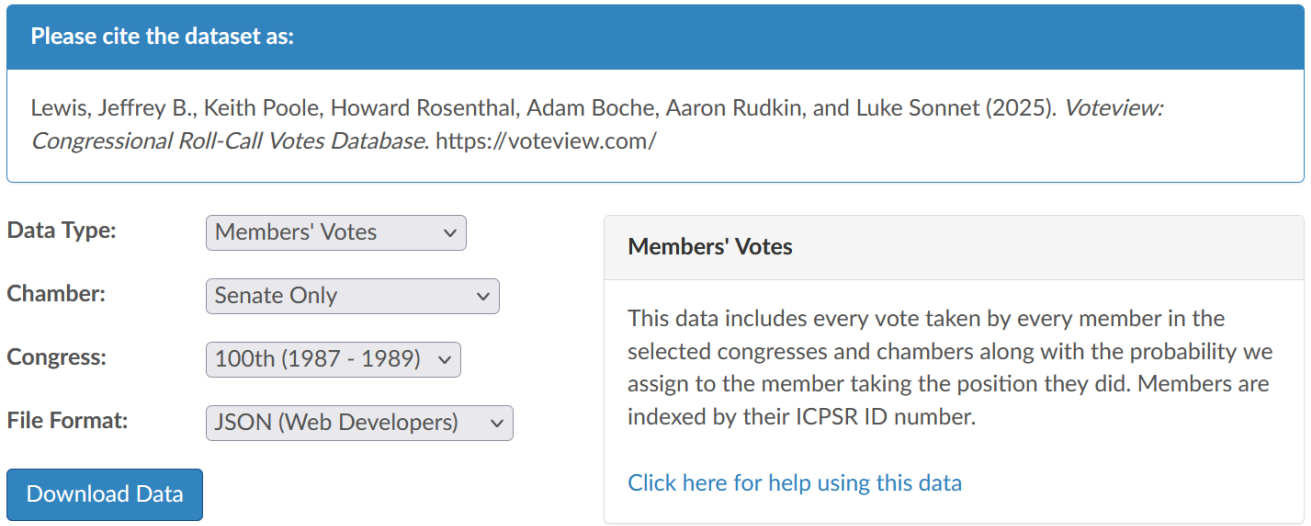
Which problems

can be solved

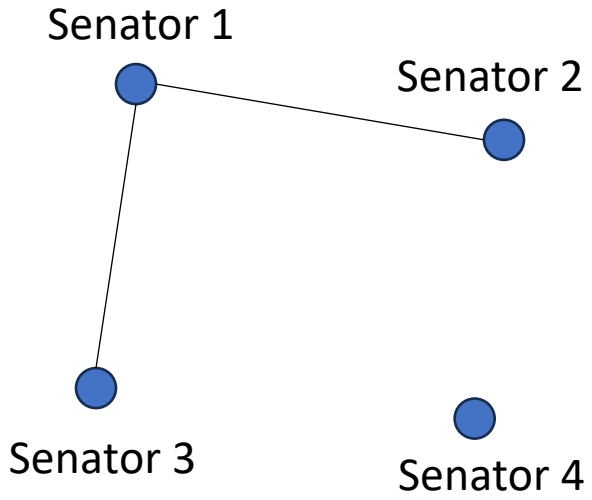
through computation?

Applying our theory

- **Question:** In the year 1988, were there 50 U.S. senators, every pair of which voted the same way more than 50% of the time?
- **Step 1:** Gather data



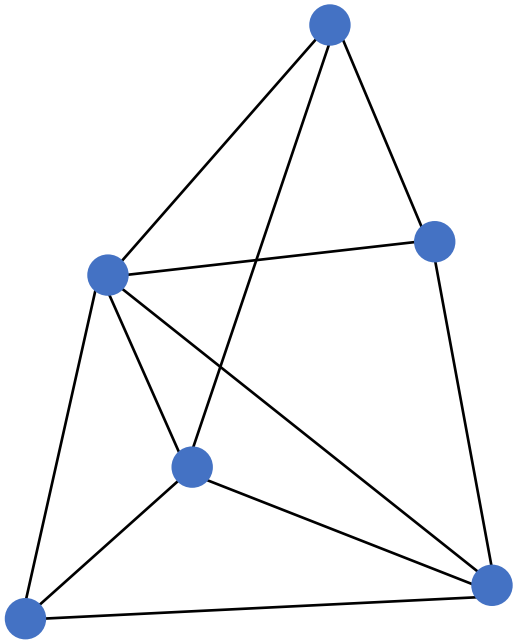
Agreement graph



- **Step 2:** Construct “agreement graph”
- Edge $\{u, v\}$ means that senators u and v agreed on most votes
- **Question:** Are there 50 vertices in this graph that are all adjacent to one another?

The clique problem

- A k-clique in a graph $G = (V, E)$ is a set $S \subseteq V$ such that $|S| = k$ and every two vertices in S are connected by an edge
- Example: This graph has a 4-clique



Which of the following statements is false?

A: Every vertex in a k-clique has degree at least $k - 1$

B: A single graph might have many k-cliques

C: If G has fewer than $\binom{k}{2}$ edges, then G does not have a k-clique

D: If every vertex has degree at least $k - 1$, then G has a k-clique

Respond at PollEv.com/whoza or text “whoza” to 22333

The clique problem

- Let $\text{CLIQUE} = \{\langle G, k\rangle : G \text{ has a } k\text{-clique}\}$
- Example: Let G be the graph with the following adjacency matrix
- Does G have a 4-clique?

	a	b	c	d	e	f	g
a	0	1	1	0	0	1	0
b	1	0	0	1	1	0	1
c	1	0	0	0	1	0	1
d	0	1	0	0	1	0	1
e	0	1	1	1	0	1	1
f	1	0	0	0	1	0	1
g	0	1	1	1	1	1	0

The clique problem

- Let $\text{CLIQUE} = \{\langle G, k\rangle : G \text{ has a } k\text{-clique}\}$
- Example: Let G be the graph with the following adjacency matrix
- Does G have a 4-clique?

Is CLIQUE decidable?

A: Yes

B: No

C: It depends on whether $\langle G\rangle$ is an adjacency matrix or adjacency list

D: It's not a language, so the question doesn't make sense

Respond at PollEv.com/whoza or text "whoza" to 22333

	a	b	c	d	e	f	g
a	0	1	1	0	0	1	0
b	1	0	0	1	1	0	1
c	1	0	0	0	1	0	1
d	0	1	0	0	1	0	1
e	0	1	1	1	0	1	1
f	1	0	0	0	1	0	1
g	0	1	1	1	1	1	0

The clique problem

- Let $\text{CLIQUE} = \{\langle G, k\rangle : G \text{ has a } k\text{-clique}\}$
- **Claim:** CLIQUE is decidable
- **Proof sketch:** Given $\langle G, k\rangle$ where $G = (V, E)$, try all possible subsets $S \subseteq V$
  - Check whether $|S| = k$
  - Check whether $\{u, v\} \in E$ for every $u, v \in S$ such that $u \neq v$
- If we find a k-clique, accept; otherwise, reject.

The clique problem

- **Question:** In the year 1988, were there 50 U.S. senators, every pair of which voted the same way more than 50% of the time?
- **Step 1:** Gather data ✔
- **Step 2:** Construct agreement graph ✔
- **Step 3:** Apply CLIQUE algorithm

Our algorithm is so slow that it's worthless

- **Question:** In the year 1988, were there 50 U.S. senators, every pair of which voted the same way more than 50% of the time?
- Checking all possible sets of senators would take longer than a lifetime!
- One begins to feel that CLIQUE might as well be undecidable!

Which problems

can be solved

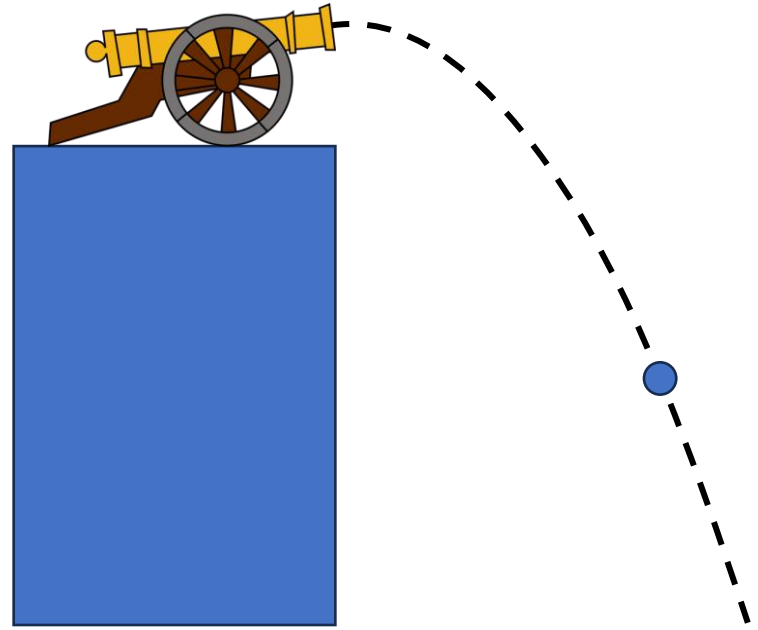
through computation?

Refining our model

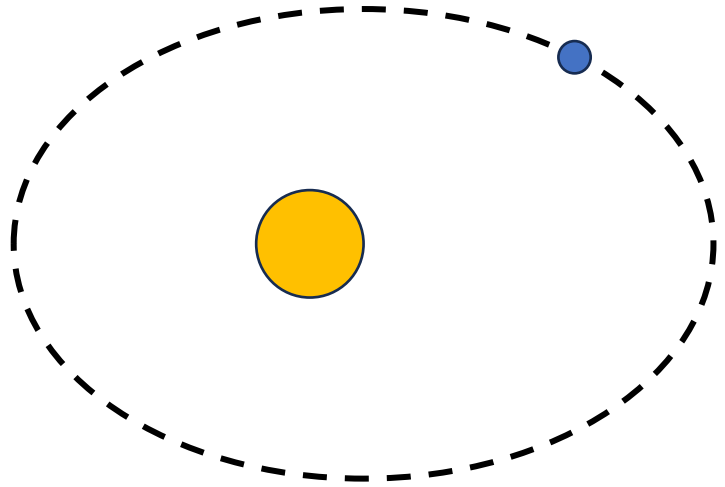
- Our model so far: Decidable vs. undecidable
- Now we will refine our model to account for the fact that we only have a limited amount of time (and other resources)
- “Complexity theory” vs. “Computability theory”

Analogy: Gravity

- Physics 101: “Gravity is a constant downward force of $9.8\ \mathrm{N/kg}$”



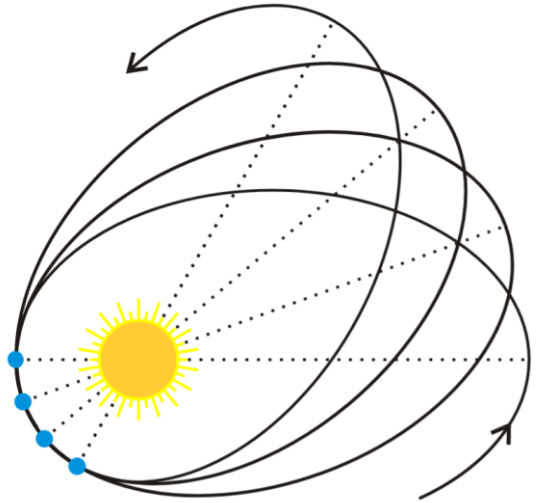
- Physics 102: Newton’s Universal Law of Gravitation: $F = G \cdot \frac{m_1 \cdot m_2}{r^2}$



Theory vs. practice

- Disclaimer: Our theoretical model will still not be perfectly accurate!
- Occasionally, we might categorize a problem as “tractable” even though it is not actually “solvable in practice”
- Other times, we might categorize a problem as “intractable” even though it is actually “solvable in practice”

Theory vs. practice



- Physics analogy: Newton's Universal Law of Gravitation is great, but it does not correctly predict Mercury's motion around the sun!
- "…all models are wrong, but some are useful." –George Box
- Even though our model of tractability will not be 100% accurate, it will still give us real insights into the nature of computation

Time complexity

- Let M be a Turing machine
- The time complexity of M is a function $T_M: \mathbb{N} \to \mathbb{N}$ defined as follows:

$$T_M(n) = \max_{w \in \{0,1\}^n} (\text{running time of } M \text{ on } w)$$

- We are focusing on the worst-case n-bit input

Scaling behavior

- We will mainly focus on the limiting behavior of $T_M(n)$ as $n \to \infty$
- How "quickly" does the time complexity $T_M(n)$ increase when we increase the input length n?

Asymptotic analysis

Asymptotic analysis

- Two possible time complexities:

$$T_1(n) = 3n^2 + 14$$

$$T_2(n) = 2n^2 + 64n + \lceil \sqrt{n} \rceil$$

- When n is large, the leading $C \cdot n^2$ term dominates
- We will ignore the low-order terms and the leading coefficient C
- We focus on the n^2 part (“quadratic time”)

Big-O notation

- $3n^2 + 14$ and $2n^2 + 64n + \lceil\sqrt{n}\rceil$ are both "$O(n^2)$"
- More generally, let $T, f: \mathbb{N} \to [0, \infty)$ be any two functions
- **Definition:** We say that T is $O(f)$ if there exist $C, n_* \in \mathbb{N}$ such that for every $n > n_*$, we have $T(n) \leq C \cdot f(n)$
- Notation: $T \in O(f)$ or $T \leq O(f)$ or $T = O(f)$

Big-O notation examples

- $3n^2 + 14$ is $O(n^2)$
- $3n^2 + 14$ is $O(n^2 + n)$
- $3n^2 + 14$ is $O(n^3)$
- $3n^2 + 14$ is not $O(n^{1.9})$

Big-Ω and big-Θ

- Let $T, f: \mathbb{N} \to [0, \infty)$ be any two functions
- We say that T is $\Omega(f)$ if there exist $c \in (0, 1)$ and $n_* \in \mathbb{N}$ such that for every $n > n_*$, we have $T(n) \geq c \cdot f(n)$
- We say that T is $\Theta(f)$ if T is $O(f)$ and T is $\Omega(f)$

Big-Ω and big-Θ examples

- $0.1n^2 + 14$ is $\Omega(n^2)$ and $\Omega(n)$, but not $\Omega(n^3)$
- $0.1n^2 + 14$ is $\Theta(n^2)$ and $\Theta(n^2 + 2n^{1.4})$, but not $\Theta(n)$

Let $T(n) = 2^{3n+4}$. Which of the following statements is false?

A: $T(n)$ is $\Omega(2^n)$

B: $T(n)$ is $2^{\Theta(n)}$

C: $T(n)$ is $\Theta(2^{3n})$

D: $T(n)$ is $O(2^n)$

Respond at PollEv.com/whoza or text “whoza” to 22333

Little-o notation

- Let $T, f: \mathbb{N} \to [0, \infty)$ be any two functions
- We say that T is $o(f)$ if for every $c \in (0, 1)$, there exists $n_* \in \mathbb{N}$ such that for every $n > n_*$, we have $T(n) < c \cdot f(n)$
- Equivalent:

$$\lim_{n \to \infty} \frac{T(n)}{f(n)} = 0$$

Little-ω notation

- Let $T, f: \mathbb{N} \to [0, \infty)$ be any two functions
- We say that T is $\omega(f)$ if for every $C \in \mathbb{N}$, there exists $n_* \in \mathbb{N}$ such that for every $n > n_*$, we have $T(n) > C \cdot f(n)$
- Equivalent:

$$\lim_{n \to \infty} \frac{T(n)}{f(n)} = \infty$$

Summary of asymptotic notation

Notation	In words	Analogy
T is $o(f)$	$T(n)$ grows more slowly than $f(n)$	$<$
T is $O(f)$	$T(n)$ is at most $C \cdot f(n)$	$\leq$
T is $\Theta(f)$	$T(n)$ and $f(n)$ grow at the same rate	$=$
T is $\Omega(f)$	$T(n)$ is at least $c \cdot f(n)$	$\geq$
T is $\omega(f)$	$T(n)$ grows more quickly than $f(n)$	$>$

Note: Big-O is not just for time complexity!

- We can use asymptotic notation (big-O, etc.) any time we are trying to understand some kind of "scaling behavior"
- For example, let G be a simple undirected graph with N vertices
  - G has $O(N^2)$ edges
  - If G is connected, then G has $\Omega(N)$ edges
- Admittedly, we are especially interested in time complexity…

Exponential vs. polynomial

- We are especially interested in the distinction between a polynomial time complexity, such as $T(n) = n^2$, and an exponential time complexity, such as $T(n) = 2^n$
- We write $T(n) = \text{poly}(n)$ if there is some k such that $T(n) = O(n^k)$
- Exponentials grow much faster than polynomials!

Exponential vs. polynomial

Claim: For every constant $k \in \mathbb{N}$, we have $n^k = o(2^n)$

- **Proof:** If $n \geq k + 1$, then

$$2^n = \#\text{ subsets of } \{1, 2, \dots, n\} = \sum_{i=0}^{n} \binom{n}{i} \geq \binom{n}{k+1} \geq \left(\frac{n}{k+1}\right)^{k+1}$$

$$= \Omega\left(n^{k+1}\right)$$

$$= \omega\left(n^k\right).$$