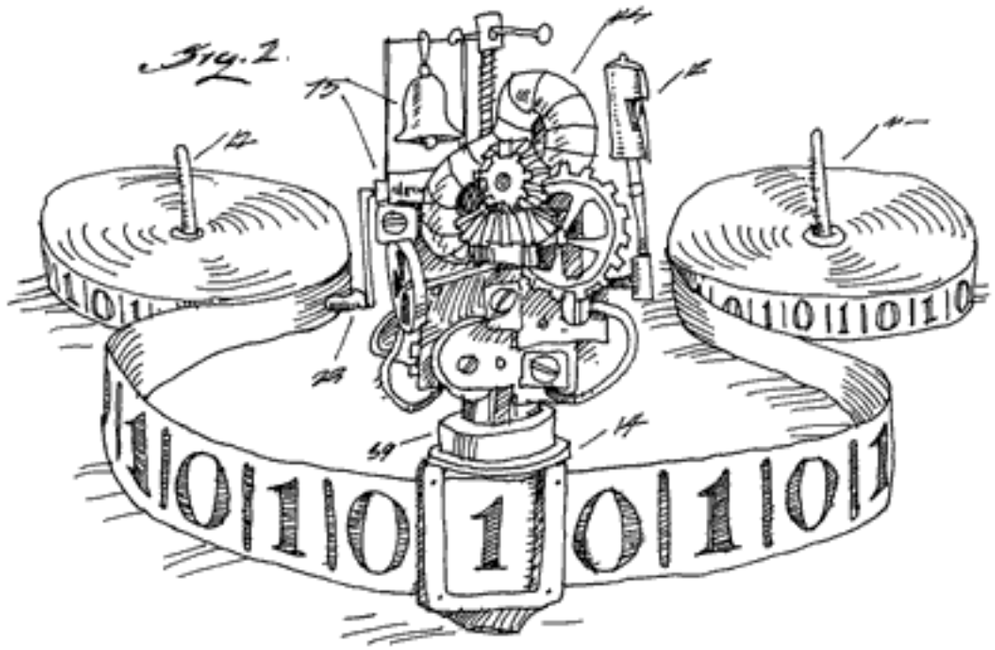
CSC4010: Introduction to Theoretical Computer Science



Lecture 10

Applying Our Theory

Question: Given a Karate Club, 20 club members each pair of which interacted outside of club meetings and karate classes?

Step 1: Gather data.

zachary: Zachary Karate Club
In zalmquist/networkdata: Lin Freeman's Network Data Collection

Description | Usage | Details | Source | Examples

Description

zachary: Two 34 by 34 matrices.

BACKGROUND These are data collected from the members of a university karate club by Wayne Zachary. The ZACHE matrix represents the presence or absence of ties among the members of the club; the ZACHC matrix indicates the relative strength of the associations (number of situations in and outside the club in which interactions occurred).

Zachary (1977) used these data and an information flow model of network conflict resolution to explain the split-up of this group following disputes among the members.

Usage

```
data(zachary)
```

Details

ZACHE: symmetric, binary.
ZACHC: symmetric, valued.

Source

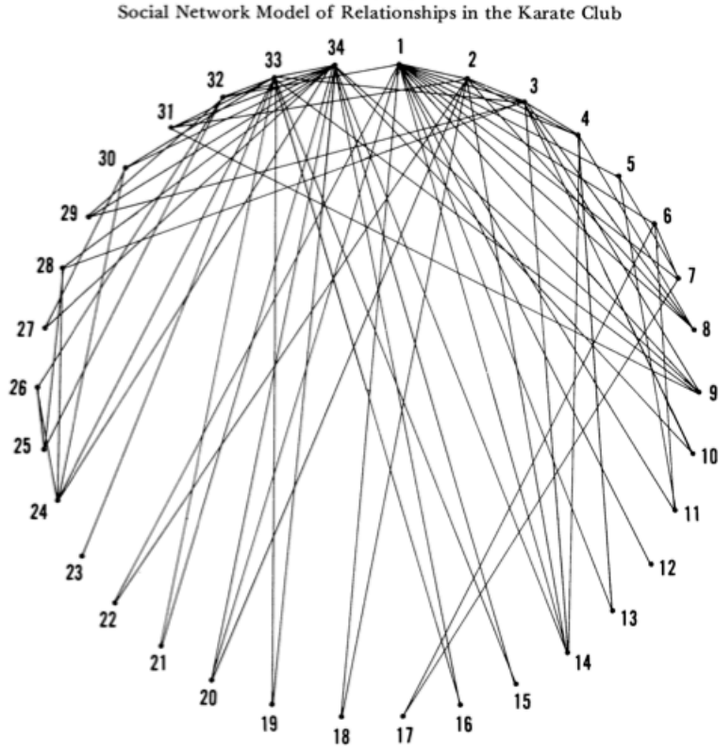
Zachary W. (1977).*An information flow model for conflict and fission in small groups.* Journal of Anthropological Research, 33, 452

Modelling the Question as a Graph

Step 2: Construct a “friendship graph”. An edge $\{u, v\}$ means that u and v met outside of club meetings and karate classes.

Question: Are there 20 vertices in this graph that are all adjacent to one another?

FIGURE 1
Social Network Model of Relationships in the Karate Club

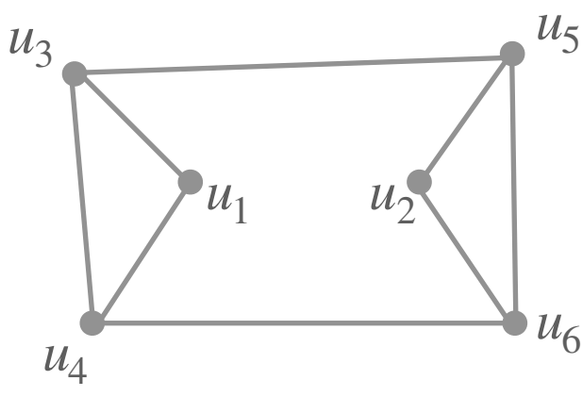


This is the graphic representation of the social relationships among the 34 individuals in the karate club. A line is drawn between two points when the two individuals being represented consistently interacted in contexts outside those of karate classes, workouts, and club meetings. Each such line drawn is referred to as an edge.

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 3-clique.



Graph G

	u_1	u_2	u_3	u_4	u_5	u_6
u_1	0	0	1	1	0	0
u_2	0	0	0	0	1	1
u_3	1	0	0	1	1	0
u_4	1	0	1	0	0	0
u_5	0	1	1	0	0	1
u_6	0	1	0	0	1	0

The Clique Problem

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

	u_1	u_2	u_3	u_4	u_5	u_6	u_7
u_1	0	1	1	0	0	1	0
u_2	1	0	0	1	1	0	1
u_3	1	0	0	0	1	0	1
u_4	0	1	0	0	1	0	1
u_5	0	1	1	1	0	1	1
u_6	1	0	0	0	1	0	1
u_7	0	1	1	1	1	1	0

The Clique Problem

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

	u_1	u_2	u_3	u_4	u_5	u_6	u_7
u_1	0	1	1	0	0	1	0
u_2	1	0	0	1	1	0	1
u_3	1	0	0	0	1	0	1
u_4	0	1	0	0	1	0	1
u_5	0	1	1	1	0	1	1
u_6	1	0	0	0	1	0	1
u_7	0	1	1	1	1	1	0

The Clique Problem

- CLIQUE $= \{\langle G, k\rangle \mid G$ has a k-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 $|S| = k$, and
  - Check whether $\{u, v\} \in E$ for every $u, v \in V$ such that $u \neq v$.
- If we find a k-clique, accept; otherwise reject.

The Clique Problem

Question: Given a Karate Club, 20 club members each pair of which interacted outside of club meetings and karate classes?

Step 1: Gather data

Step 2: Construct the “friendship graph”

Step 3: Apply CLIQUE algorithm

The algorithm is so slow it is worthless

Question: Given a Karate Club, 20 club members each pair of which interacted outside of club meetings and karate classes?

Checking all possible sets of members of could take 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

- We only have a limited amount of time (and other resources)

“Complexity Theory” versus “Computability Theory”

Time Complexity

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

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

- We focus on the worst-case n-bit input.

Scaling Behaviour

- We focus on the limiting behaviour of $T_M(n)$ as $n \to \infty$.
- How “quickly” does the running time increase as we consider larger and larger inputs.

Asymptotic Analysis

- Two time complexities:

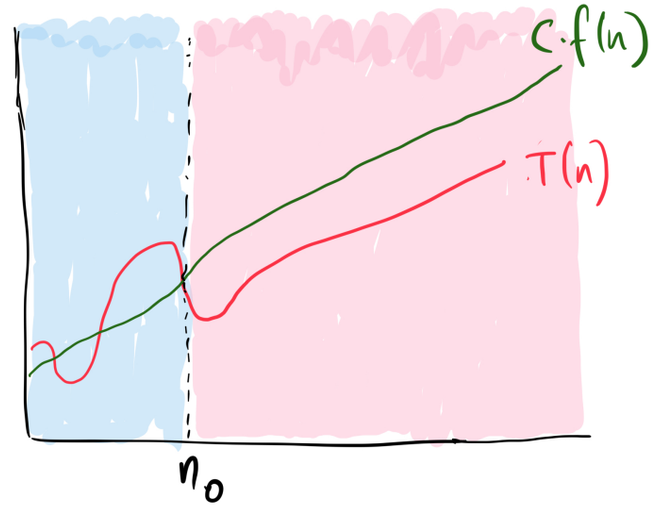
$$T_1 = 3n^2 + 14$$
$$T_2(n) = 2n^2 + 64n + \lfloor n \rfloor$$

- When n is large, the leading $C \cdot n^2$ term dominates.

- We will ignore the low-order terms and the leading coefficients.

- We focus on the n^2 part (“quadratic time”)

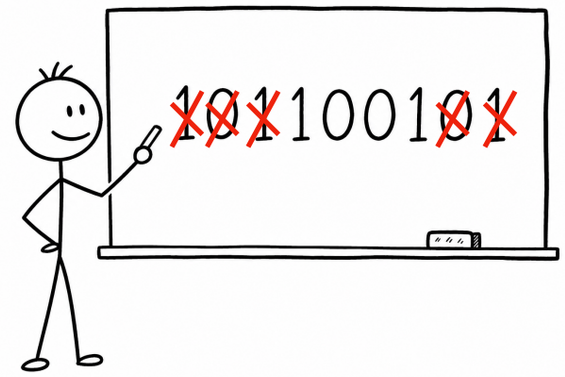
Big-O Notation



- Let $T, f : \mathbb{N} \to [0, \infty)$ be any two functions.
- **Definition:** We say $T(n) = O(f(n))$ if there exist positive constants c, n_0 such that $T(n) \leq c \cdot f(n)$ for all $n \geq n_0$.
- **Notation:** $T \in O(f)$ or $T = O(f)$.

- **Examples:**
  - $3n^2 + 14$ is $O(n^2)$
  - $3n^2 + 14$ is $O(n^2 + n)$
  - $3n^2 + 14$ is $O(n^3)$
  - $\log n$ is $O(n^2)$
  - $3n^2 + 14$ is not $O(n^{1.9})$

Example: Palindromes



Claim: There exists a Turing Machine that decides palindrome with time complexity $O(n^2)$

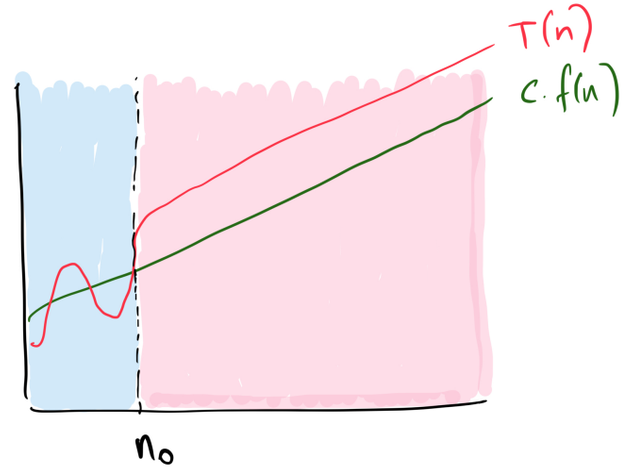
- **Proof:** Recall our Turing Machine that decides PALINDROME.
- It makes about $n/2$ back and forth passes over the input to cross out symbols.
- In each pass, it takes $O(n)$ steps.
- Total time complexity is $n/2 \cdot O(n) = O(n^2)$.

Big-Ω Notation

- Is there a faster Turing machine that decides PALINDROME?
- **Answer:** No. We will use Big-Ω notation to make this precise.

- Let $T, f : \mathbb{N} \to [0,\infty)$ be any two functions.

- **Definition:** We say $T(n) = \Omega(f(n))$ if there exist positive constants c, n_0 such that $T(n) \geq c \cdot f(n)$ for all $n \geq n_0$.
- **Notation:** $T \in \Omega(f)$ or $T = \Omega(f)$.
- **Example:** $3n^2$ is $\Omega(n)$ and $\Omega(n^2)$, but not $\Omega(n^3)$.



Palindromes: Time Complexity Lower Bound

- Let M be a one-tape Turing Machine:

> **Claim:** If M decides PALINDROME, then M has time complexity $\Omega(n^2)$.

- **Proof:** Omitted for now. But maybe as a bonus problem in HW.

Palindromes: On two Tape Machines

Claim: There exists a two-tape Turing Machine that decides PALINDROME in time complexity $O(n)$.

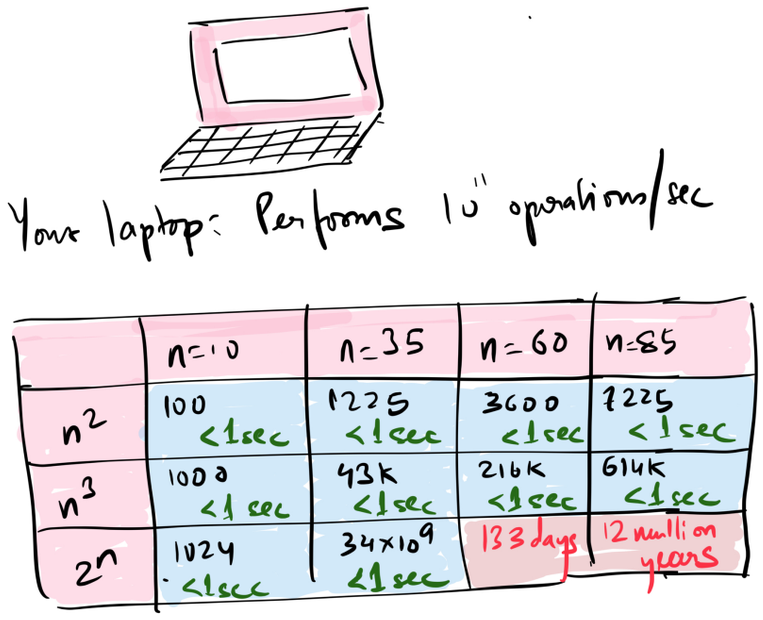
- **Proof Sketch:**
  - Copy the input to the second tape.
  - Move the head of second tape to the end of the input.
  - Move head 1 from left to right and head 2 from right to left, and compare.

Multi-Tape vs Two-Tape TMs

- Multi-tape Turing Machines can decide PALINDROME in $O(n)$ time.
- But, one-tape Turing Machine takes $\Omega(n^2)$ time.
- **Question:** So, these two models are not equivalent afterall?

Exponential vs. Polynomial

- In this module, we are not concerned with the distinction between $O(n)$ and $O(n^2)$ time. We are happy with either.
- **Our focus:** The distinction between polynomial time such as a $T(n) = n^2$ and exponential time such as $T(n) = 2^n$.
- We write $T(n) = \text{poly}(n)$ if there exists a constant c such that $T(n) = O(n^c)$.
- Exponentials grow much faster than polynomials.



- Make this precise using little - o and little - ω notation.

Little-o Notation

- Let $T, f : \mathbb{N} \to [0,\infty)$ be any two functions.
- **Definition:** We say $T(n) = o(f(n))$ if for every $c \in (0,1)$, there exists $n_0 \in \mathbb{N}$ such that for every $n > n_0$, $T(n) < c \cdot f(n)$.
- **Equivalently:** $\lim_{n\to\infty} \dfrac{T(n)}{f(n)} = 0.$

Little-ω Notation

- Let $T, f : \mathbb{N} \to [0, \infty)$ be any two functions.
- **Definition:** We say $T(n) = \omega(f(n))$ if for every $c \in \mathbb{N}$, there exists $n_0 \in \mathbb{N}$ such that for every $n > n_0$, $T(n) > c \cdot f(n)$.
- **Equivalently:** $\lim_{n \to \infty} \frac{T(n)}{f(n)} = \infty$

Exponential vs Polynomial

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

Big-Θ Notation

- Let $T, f : \mathbb{N} \to [0,\infty)$ be any two functions.
- We say $T(n) = \Theta(f(n))$ if $T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$.
- **Example:** $3n^2 + 14$ is $\Theta(n^2)$ but not $\Theta(n)$. Let $T, f : \mathbb{N} \to [0,\infty)$ be any two functions.

Summary

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

Deciding a Language in Time T

- Let $Y \subseteq \{0,1\}^*$ and let $T : \mathbb{N} \to [0,\infty)$ be a function.
- **Definition:** We say that Y is can be decided in time T if there exists a one-tape Turing Machine M such that
  - M decides Y,
  - For every $n \in \mathbb{N}$ and $w \in \{0,1\}^n$, the running time of M on w is at most $T(n)$.

The Complexity Class P

- **Definition:** For any function $T : \mathbb{N} \to [0, \infty)$, we define

$$\text{TIME}(T) = \{ Y \subseteq \{0,1\}^* \mid Y \text{ can be decided in time } T \}.$$

- **Definition:**

$$P = \{ Y \subseteq \{0,1\}^* \mid Y \text{ can be decided in time poly}(n) \}$$
$$= \bigcup_{k=1}^{\infty} \text{TIME}(n^k)$$

- "Polynomial Time"

P: Our model of Tractability

- Let $Y \subseteq \{0,1\}^*$
- If $Y \in P$, then Y is “tractable”.
- If $Y \notin P$, then Y is “intractable”.
- Is this a good definition of tractability? Is this robust across models of computation? Are there problems which are decidable but not in P .

Multi-tape Turing Machines

- Let $Y \subseteq \{0,1\}^*$, let k be any positive integer, and let $T : \mathbb{N} \to \mathbb{N}$

Lemma: If there exists a k-tape Turing machine that decides Y with time complexity $T(n)$, then there exists a 1-tape Turing machine that decides Y with time complexity $O(T(n)^2)$.

Multi-tape Turing Machines

- Let $Y \subseteq \{0,1\}^*$, let k be any positive integer, and let $T : \mathbb{N} \to \mathbb{N}$

Lemma: If there exists a k-tape Turing machine that decides Y with time complexity $T(n)$, then there exists a 1-tape Turing machine that decides Y with time complexity $O(T(n)^2)$.

- **Proof:** For simplicity, assume $T(n) \geq n$.
- Recall: To simulate step i, we scan back and forth over $n + 2i$ cells of the tape.
- Simulating one step of the k-tape machine takes $O(n + T(n))$ steps.
- Overall time complexity: $O(T(n)^2)$.

