Outline

- 3-SAT
- The complexity class EXP
- Time Hierarchy Theorem
- hardness and completeness
  - an EXP-complete problem
- The class NP
  - alternate characterization of NP

2SAT

**Theorem**: There is a polynomial-time algorithm deciding 2SAT (“2SAT ∈ P”).

Proof: algorithm described on next slides.

Algorithm for 2SAT

**Claim**: formula is unsatisfiable iff there is some variable \( x \) with a path from \( x \) to \( \neg x \) and a path from \( \neg x \) to \( x \) in derived graph.

- **Proof (\( \Leftarrow \))**
  - edges represent implication \( \Rightarrow \). By transitivity of \( \Rightarrow \), a path from \( x \) to \( \neg x \) means \( x \Rightarrow \neg x \), and a path from \( \neg x \) to \( x \) means \( \neg x \Rightarrow x \).

Algorithm for 2SAT

- **Proof (\( \Rightarrow \))**
  - to construct a satisfying assign. (if no \( x \) with a path from \( x \) to \( \neg x \) and a path from \( \neg x \) to \( x \)):
    - pick unassigned literal \( s \) with no path from \( s \) to \( \neg s \)
    - assign it TRUE, as well as all nodes reachable from it; assign negations of these literals FALSE
    - note: path from \( s \) to \( t \) and \( s \) to \( \neg t \) implies path from \( \neg t \) to \( \neg s \) and \( t \) to \( \neg s \), so \( s \) already assigned at that point.
Algorithm for 2SAT

- **Algorithm:**
  - build derived graph
  - for every pair $x$, $\neg x$ check if there is a path from $x$ to $\neg x$ and from $\neg x$ to $x$ in the graph
- **Running time of algorithm (input length $n$):**
  - $O(n)$ to build graph
  - $O(n)$ to perform each check
  - $O(n)$ checks
  - running time $O(n^2)$. 2SAT $\in P$.

Another puzzle

- Find an efficient algorithm to solve the following problem.
- **Input:** sequence of *triples* of symbols
  - e.g. $(A, b, C)$, $(E, D, b)$, $(d, A, C)$, $(c, b, a)$
- **Goal:** determine if it is possible to circle at least one symbol in each *triple* without circling upper and lower case of same symbol.

3SAT

- This is a disguised version of the language $3SAT = \{\text{formulas in Conjunctive Normal Form with 3 literals per clause for which there exists a satisfying truth assignment}\}$
  - e.g. $(A, b, C)$, $(E, D, b)$, $(d, A, C)$, $(c, b, a)$
    $(x_1V \neg x_2V x_3)\land (x_3V x_2V \neg x_3)\land (\neg x_1V x_3V x_2)\land (\neg x_3V \neg x_2V \neg x_1)$
- observe that this language is in $TIME(2^n)$

Time Complexity

**Key definition:** "P" or "polynomial-time" is $P = \bigcup_{k \geq 1} TIME(n^k)$

**Definition:** "EXP" or "exponential-time" is $EXP = \bigcup_{k \geq 1} TIME(2^{n^k})$

- Note: $P \subseteq EXP$.
- We have seen 3SAT $\in EXP$.
  - does not rule out possibility that it is in $P$
- Is $P$ different from $EXP$?

Time Hierarchy Theorem

**Theorem:** for every proper complexity function $f(n) \geq n$:

$TIME(f(n)) \not\subseteq TIME(f(2n)^3)$.

- Note: $P \subseteq TIME(2^n) \not\subseteq TIME(2^{(2n)^3}) \subseteq EXP$
- Most natural functions (and $2^n$ in particular) are proper complexity functions.
  We will ignore this detail in this class.
**Time Hierarchy Theorem**

**Theorem:** for every proper complexity function $f(n) \geq n$:

$$\text{TIME}(f(n)) \subsetneq \text{TIME}(f(2n)^3).$$

- **Proof idea:**
  - Use diagonalization to construct a language that is not in $\text{TIME}(f(n))$.
  - Constructed language comes with a TM that decides it and runs in time $f(2n)^3$.

**Recall proof for Halting Problem**

- **Inputs:** Turing Machines
  - **Box** $(M, x)$: does $M$ halt on $x$?

**Proof of Time Hierarchy Theorem**

- **Proof:**
  - **Claim:** there is a TM SIM that decides
    $$\{ <M, x> : M \text{ accepts } x \text{ in } \leq f(|x|) \text{ steps} \}$$
  - **Proof sketch:** SIM has 4 work tapes
    - Contents and "virtual head" positions for $M$’s tapes
    - $M$’s transition function and state
    - $f(|x|)$’s used as a clock
    - Scratch space

- **Proof (continued):**
  - Suppose $M$ in $\text{TIME}(f(n))$ decides $L(D)$
    - $M(<M>) = \text{SIM}(<M, <M>)) \neq D(<M>)$
    - But $M(<M>) = D(<M>)$
  - Contradiction.
Proof of Time Hierarchy Theorem

- Proof sketch (continued): 4 work tapes
  - contents and "virtual head" positions for M's tapes
  - M's transition function and state
  - \( f(|x|) \) "s used as a clock
  - scratch space
- Initialize tapes
- Simulate step of M, advance head on tape 3; repeat.
- Can check running time is as claimed.

So far…

- We have defined the complexity classes \( P \) (polynomial time), \( \text{EXP} \) (exponential time)

Poly-time reductions

- Type of reduction we will use:
  - "many-one" poly-time reduction (commonly)
  - "mapping" poly-time reduction (book)

Definition: \( A \leq_p B \) ("A reduces to B") if there is a poly-time computable function \( f \) such that for all \( w \)

\[ w \in A \iff f(w) \in B \]

- As before, condition equivalent to:
  - YES maps to YES and NO maps to NO
- As before, meaning is:
  - B is at least as "hard" (or expressive) as A

Theorem: If \( A \leq_p B \) and \( B \in P \) then \( A \in P \).

Proof:
- A poly-time algorithm for deciding A:
- On input w, compute \( f(w) \) in poly-time.
- Run poly-time algorithm to decide if \( f(w) \in B \)
- If it says "yes", output "yes"
- If it says "no", output "no"
Example

• 2SAT = \{CNF formulas with 2 literals per clause for which there exists a satisfying truth assignment\}
• L = \{directed graph G, and list of pairs of vertices \((u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\), such that there is no \(i\) for which \(u_i\) is reachable from \(v_i\) in G and \(v_i\) is reachable from \(u_i\) in G\}
• We gave a poly-time reduction from 2SAT to L.
• determined that 2SAT \(\in\) P from fact that L \(\in\) P

Hardness and completeness

• Reasonable that can efficiently transform one problem into another.

• Surprising:
  – can often find a special language L so that every language in a given complexity class reduces to L!
  – powerful tool

Hardness and completeness

• Recall:
  – a language L is a set of strings
  – a complexity class C is a set of languages

Definition: a language L is C-hard if for every language A \(\in\) C, A poly-time reduces to L; i.e., A \(\leq_P\) L.
meaning: L is at least as “hard” as anything in C

Definition: a language L is C-complete if L is C-hard and L \(\in\) C
meaning: L is a “hardest” problem in C

An EXP-complete problem

• Version of ATM with a time bound:
  ATM_B = \{\langle M, x, m \rangle : M is a TM that accepts x within at most m steps\}

Theorem: ATM_B is EXP-complete.

Proof:
  – what do we need to show?

An EXP-complete problem

• ATM_B = \{\langle M, x, m \rangle : M is a TM that accepts x within at most m steps\}
• Proof that ATM_B is EXP-complete:
  – Part 1. Need to show ATM_B \(\in\) EXP.
    • simulate M on x for m steps; accept if simulation accepts; reject if simulation doesn’t accept.
    • running time \(m^{O(1)}\).
    • \(n = \) length of input \(\geq \log m\)
    • running time \(\leq m = 2^{\log m} \leq 2^{O(1)}\)
An EXP-complete problem

- ATMₖ = {<M, x, m> : M is a TM that accepts x within at most m steps}
- Proof that ATM₂ is EXP-complete:
  - Part 2. For each language A ∈ EXP, need to give poly-time reduction from A to ATM₂.
  - for a given language A ∈ EXP, we know there is a TM Mₐ that decides A in time g(n) ≤ 2ⁿ for some k.
  - what should reduction f(w) produce?

Proof that ATM₂ is EXP-complete:
- f(w) = <Mₐ, w, m> where m = 2ⁿk
- is f(w) poly-time computable?
  - hardcoded Mₐ and k…
  - YES maps to YES?
    - w ∈ A ⇒ <Mₐ, w, m> ∈ ATM₂
  - NO maps to NO?
    - w ∉ A ⇒ <Mₐ, w, m> ∉ ATM₂

An EXP-complete problem

- A C-complete problem is a surrogate for the entire class C.
- For example: if you can find a poly-time algorithm for ATM₂ then there is automatically a poly-time algorithm for every problem in EXP (i.e., EXP = P).
- Can you find a poly-time alg for ATM₂?

An EXP-complete problem

- Can you find a poly-time alg for ATM₂?
  - NO! we showed that P ⊈ EXP.
  - ATM₂ is not tractable (intractable).