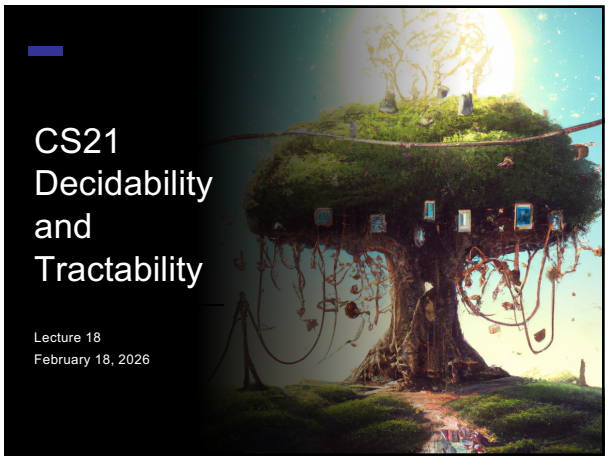


1

Grades so far

- An idea of eventual scale:
- 2026 mean: 90.6
- 2025 mean: 89.4; median 90.0
- 2024 mean: 85.5; median 87.0
- 2023 mean 80.5; median 81.4
- 2022 mean 80.9; median 83.6

2025

min	max	grade
98.0	100.0	A+
93.0	98.0	A
90.0	93.0	A-
87.0	90.0	B+
84.0	87.0	B
80.0	84.0	B-
77.0	80.0	C+
73.0	77.0	C
70.0	73.0	C-
67.0	70.0	D+
63.0	67.0	D
0.0	63.0	E/F

2024

min	max	grade
97.0	100.0	A+
93.0	97.0	A
88.0	93.0	A-
85.0	88.0	B+
81.0	85.0	B
78.0	81.0	B-
74.5	78.0	C+
70.0	74.5	C
65.0	70.0	C-
60.0	65.0	D+
55.0	60.0	D
0.0	55.0	E/F

2023

min	max	grade
97.0	100.0	A+
92.0	97.0	A
87.0	92.0	A-
84.0	87.0	B+
80.5	84.0	B
76.0	80.5	B-
72.5	76.0	C+
68.0	72.5	C
62.5	68.0	C-
59.0	62.5	D+
52.5	59.0	D
0.0	52.5	E/F

2022

min	max	grade
97.0	100.0	A+
92.0	97.0	A
87.0	92.0	A-
84.0	87.0	B+
80.5	84.0	B
76.0	80.5	B-
72.5	76.0	C+
68.0	72.5	C
62.5	68.0	C-
59.0	62.5	D+
54.0	59.0	D
0.0	54.0	E/F

2

2

A puzzle

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

3

A puzzle

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

4

2SAT

- This is a disguised version of the language
  2SAT = {formulas in Conjunctive Normal Form with 2 literals per clause for which there exists a satisfying truth assignment}
  – CNF = "AND of ORs"

(A, b), (E, D), (d, C), (b, a)

$(x_1 \vee \neg x_2) \wedge (x_5 \vee x_4) \wedge (\neg x_4 \vee x_3) \wedge (\neg x_2 \vee \neg x_1)$

  – satisfying truth assignment = assignment of TRUE/FALSE to each variable so that whole formula is TRUE

5

2SAT

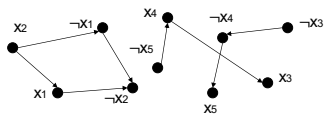
Theorem: There is a polynomial-time algorithm deciding 2SAT ("2SAT $\in$ P").

Proof: algorithm described on next slides.

6

Algorithm for 2SAT

- Build a graph with separate nodes for each literal.
  - add directed edge (x, y) iff formula includes clause (¬x ∨ y) (equiv. to x ⇒ y)



e.g. $(x_1 \vee \neg x_2) \wedge (x_5 \vee x_4) \wedge (\neg x_4 \vee x_3) \wedge (\neg x_2 \vee \neg x_1)$

7

Algorithm for 2SAT

Claim: formula is unsatisfiable iff there is some variable x with a path from x to ¬x and a path from ¬x to x in derived graph.

- Proof (⇐)
  - edges represent implication ⇒. By transitivity of ⇒, a path from x to ¬x means x ⇒ ¬x, and a path from ¬x to x means ¬x ⇒ x.

8

Algorithm for 2SAT

- Proof (⇒)
  - to construct a satisfying assign. (if no x with a path from x to ¬x and a path from ¬x to x):
    - pick unassigned literal s with no path from s to ¬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 ¬t implies path from ¬t to ¬s and t to ¬s, implies path from s to ¬s
    - note: path s to t (assigned FALSE) implies path from ¬t (assigned TRUE) to ¬s, so s already assigned at that point.

9

Algorithm for 2SAT

- Algorithm:
  - build derived graph
  - for every pair x, ¬x check if there is a path from x to ¬x and from ¬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 ∈ P.

10

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.

11

3SAT

- This is a disguised version of the language
  3SAT = {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)
  $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_5 \vee x_4 \vee \neg x_2) \wedge (\neg x_4 \vee x_1 \vee x_3) \wedge (\neg x_3 \vee \neg x_2 \vee \neg x_1)$
- observe that this language is in $TIME(2^n)$

12

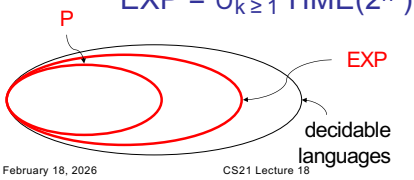
Time Complexity

Key definition: "P" or "polynomial-time" is

$P = \cup_{k \geq 1} TIME(n^k)$

Definition: "EXP" or "exponential-time" is

$EXP = \cup_{k \geq 1} TIME(2^{n^k})$



13

EXP

$P = \cup_{k \geq 1} TIME(n^k)$

$EXP = \cup_{k \geq 1} TIME(2^{n^k})$

- 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?

14

Time Hierarchy Theorem

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

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

- Note: $P \subseteq TIME(2^n) \subsetneq TIME(2^{(2n)3}) \subseteq EXP$
- Most natural functions (and 2^n in particular) are proper complexity functions. We will ignore this detail in this class.

15

Time Hierarchy Theorem

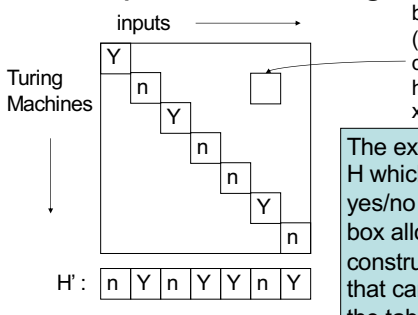
Theorem: for every proper complexity function $f(n) \geq n$:

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

- Proof idea:
  - use diagonalization to construct a language that is not in TIME(f(n)).
  - constructed language comes with a TM that decides it and runs in time $f(2n)^3$.

16

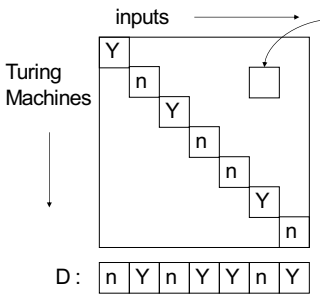
Recall proof for Halting Problem



The existence of H which tells us yes/no for each box allows us to construct a TM H' that cannot be in the table.

17

Proof of Time Hierarchy Theorem



- TM SIM tells us yes/no for each box **in time g(n)**
- rows include all of **TIME(f(n))**
- construct TM D running in time g(2n) that is not in table

18

Proof of Time Hierarchy Theorem

- Proof:
  - SIM is TM deciding language
    { <M, x> : M accepts x in ≤ f(|x|) steps }
  - Claim: SIM runs in time $g(n) = f(n)^3$.
  - define new TM D: on input <M>
    - if SIM accepts <M, <M>>, reject
    - if SIM rejects <M, <M>>, accept
  - D runs in time g(2n)

19

Proof of Time Hierarchy Theorem

- Proof (continued):
  - suppose M in **TIME(f(n))** decides L(D)
    - M(<M>) = SIM(<M, <M>>) ≠ D(<M>)
    - but M(<M>) = D(<M>)
  - contradiction.

20

Proof of Time Hierarchy Theorem

- Claim: there is a TM SIM that decides
  {<M, x> : M accepts x in ≤ f(|x|) steps}
  and runs in time $g(n) = f(n)^3$.
- 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

21

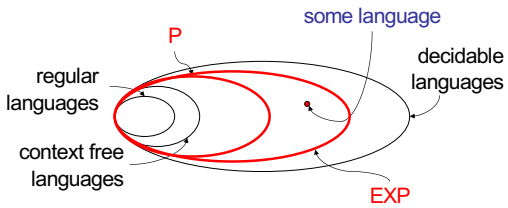
Proof of Time Hierarchy Theorem

- 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.

22

So far…

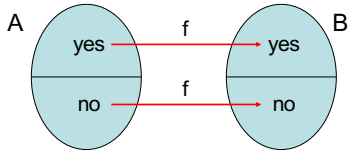
- We have defined the complexity classes P (polynomial time), EXP (exponential time)



23

Poly-time reductions

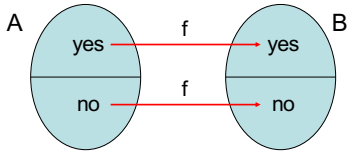
- Type of reduction we will use:
  - "many-one" poly-time reduction (commonly)
  - "mapping" poly-time reduction (book)



reduction from language A to language B

24

Poly-time reductions



- function f should be poly-time computable

Definition: f : $\Sigma^* \rightarrow \Sigma^*$ is poly-time computable if for some $g(n) = n^{O(1)}$ there exists a g(n)-time TM M_f such that on every $w \in \Sigma^*$, M_f halts with f(w) on its tape.

25

Poly-time reductions

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

$$w \in A \Leftrightarrow f(w) \in B$$

- 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

26

Poly-time reductions

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"

27

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

28

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

29

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

30

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-complete if L is C-hard and L ∈ C

meaning: L is a "hardest" problem in C