CS21
Decidability and Tractability

Lecture 17

February 12, 2024

1

Euclid's Algorithm

on input <x, y>:
- (1) repeat until y = 0
  - (2) set x = x mod y
  - (3) swap x, y
- x is the GCD(x, y). If x = 1, accept; otherwise reject

- every 2 times through loop, (x, y) each reduced by ½
- loops ≤ 2max{$\log_2 x$, $\log_2 y$} = O(n = |<x, y>|); poly time for each loop

Claim: value of x reduced by ½ at every execution of (2) except possibly first one.

Proof:
- after (2) x < y
- after (3) x > y
- if x/2 ≥ y, then x mod y < y ≤ x/2
- if x/2 < y, then x mod y = x – y < x/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

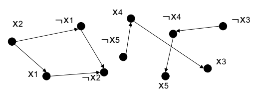
Theorem: There is a polynomial-time algorithm deciding 2SAT ("2SAT ∈ P").

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 $(\neg x \vee y)$ (equiv. to $x \Rightarrow y$)



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

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 $x \Rightarrow \neg x$, and a path from ¬x to x means $\neg x \Rightarrow x$.

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.

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.

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 = {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 $TIME(2^n)$

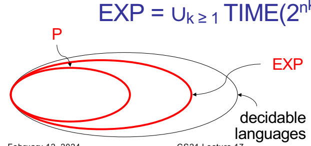
Time Complexity

Key definition: “P” or “polynomial-time” is

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

Definition: “EXP” or “exponential-time” is

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



February 12, 2024 CS21 Lecture 17 13

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?

February 12, 2024 CS21 Lecture 17 14

14

Time Hierarchy Theorem

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

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

- 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.

February 12, 2024 CS21 Lecture 17 15

15

Time Hierarchy Theorem

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

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

- 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 $f(2n)^3$.

February 12, 2024 CS21 Lecture 17 16

16