CS 401

NP-Complete / Final Review

Xiaorui Sun

NP Completeness

Complexity Theorists Approach: We don't know how to prove any problem in NP is hard. So, let's find hardest problems in NP.

NP-complete: A problem Y in NP with the property that for every problem X in NP, $X \leq_p Y$.

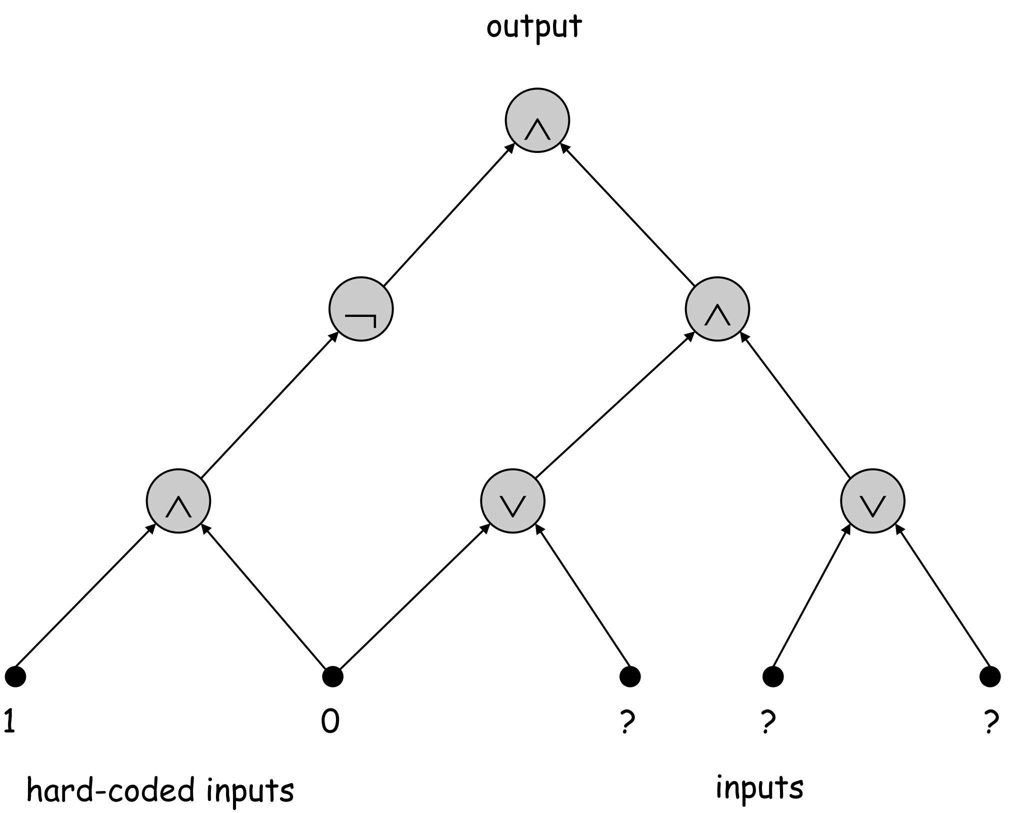
The "First" NP-Complete Problem

CIRCUIT-SAT. Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]

Read KT 8.4 for the proof

yes: 1 0 1



Establishing NP-Completeness

Remark. Once we establish first "natural" NP-complete problem, others fall like dominoes.

Recipe to establish NP-completeness of problem Y.

- Step 1. Show that Y is in NP.
- Step 2. Choose a known NP-complete problem X.
- Step 3. Prove that $X \leq_p Y$.

Justification. If X is an NP-complete problem, and Y is a problem in NP with the property that $X \leq_P Y$ then Y is NP-complete.

Pf. Let W be any problem in NP. Then $W \underbrace{\leq_P}_{\text{by definition of NP-complete}} X \underbrace{\leq_P}_{\text{by assumption}} Y$.

- By transitivity, $W \leq_P Y$.
- Hence Y is NP-complete. ▪

3-SAT is NP-Complete

SAT: Given CNF formula Φ, does it have a satisfying truth assignment?

Ex: $\left(\overline{x_1} \lor x_2 \lor x_3\right) \land \left(x_1 \lor \overline{x_2} \lor x_3\right) \land \left(x_2 \lor x_3\right) \land \left(\overline{x_1} \lor \overline{x_2} \lor \overline{x_3}\right)$

Yes: x_1 = true, x_2 = true x_3 = false.

3-SAT: SAT where each clause contains exactly 3 literals.

Theorem. 3-SAT is NP-complete.

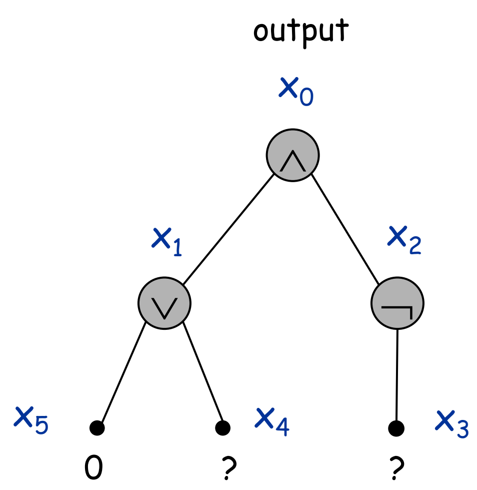
Based on CIRCUIT-SAT is NP complete, how to show 3-SAT is NP-complete?

3-SAT is NP-Complete

Theorem. 3-SAT is NP-complete.

Pf. Suffices to show that CIRCUIT-SAT $\leq_P$ 3-SAT since 3-SAT is in NP.

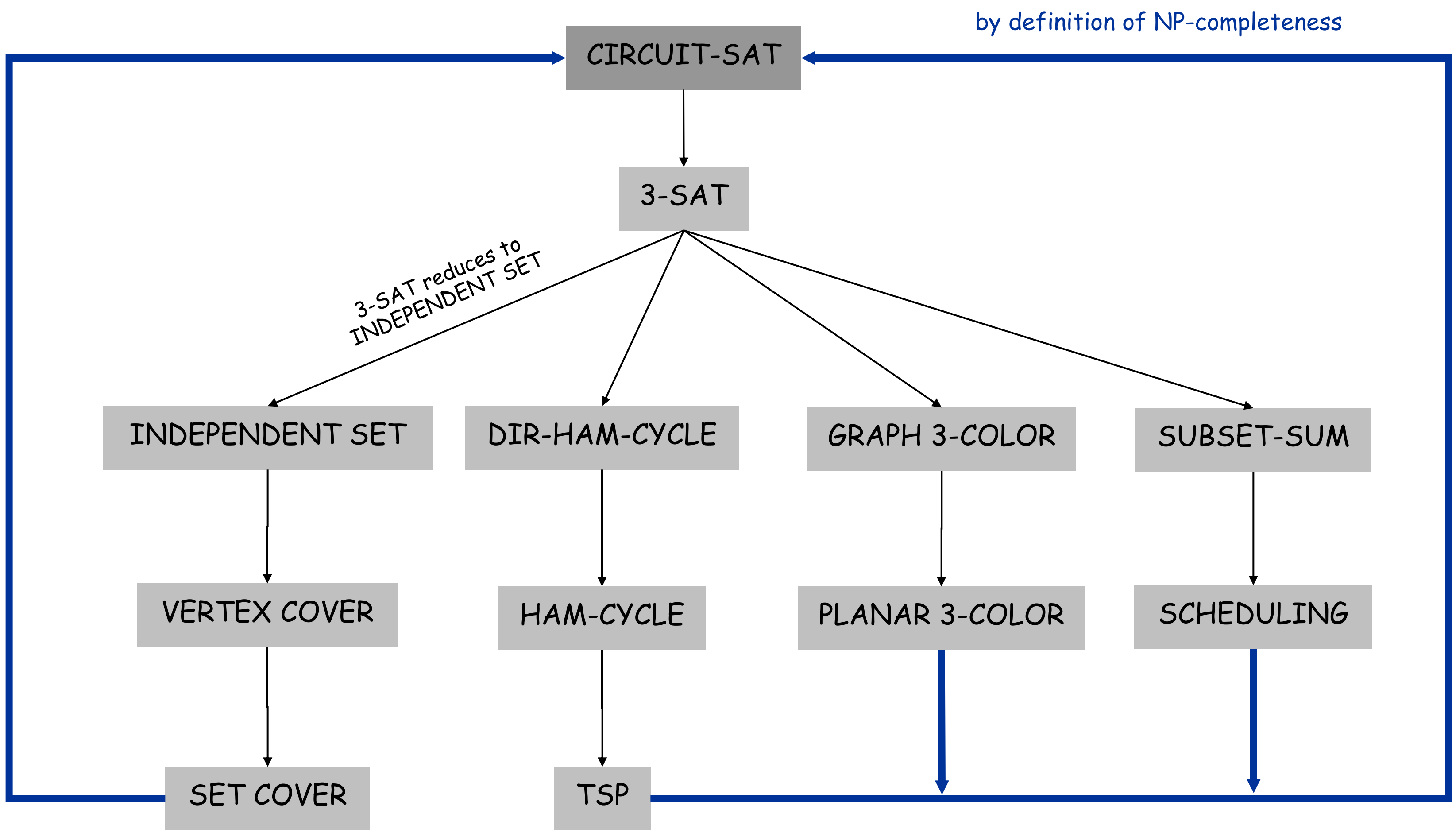
- Let K be any circuit.
- Create a 3-SAT variable x_i for each circuit element i.
- Make circuit compute correct values at each node:
  - $x_2 = \neg\, x_3 \Rightarrow$ add 2 clauses: $x_2 \vee x_3$, $\overline{x_2} \vee \overline{x_3}$
  - $x_1 = x_4 \vee x_5 \Rightarrow$ add 3 clauses: $x_1 \vee \overline{x_4}$, $x_1 \vee \overline{x_5}$, $\overline{x_1} \vee x_4 \vee x_5$
  - $x_0 = x_1 \wedge x_2 \Rightarrow$ add 3 clauses: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$



- Hard-coded input values and output value.
  - $x_5 = 0 \Rightarrow$ add 1 clause: $\overline{x_5}$
  - $x_0 = 1 \Rightarrow$ add 1 clause: x_0
- Final step: turn clauses of length < 3 into clauses of length exactly 3. ▪

NP-Completeness

Observation. All problems below are NP-complete and polynomial reduce to one another!



Summary

To show a problem is in NP:

- Make sure it is a decision problem
- And give a polynomial time certifier

Recipe to show a problem Y is NP-Complete

- Step 1. Show that Y is in NP.
- Step 2. Choose a known NP-complete problem X.
- Step 3. Prove that $X \le_p Y$.

Final Exam Review

Stuff

Teaching evaluation

- Extra 1% score for all the students if overall response rate >= 80%
- May improve the final grade
- Teaching evaluation will close on Sunday April 28

My office hour next week will be Wednesday May 1 3pm– 5pm

TA’s office hour next week will be schedule to another time (Monday or Thursday)

Study materials

Homework 5 will be helpful for you to prepare final exam

A sample final exam will be released later today

Additional algorithm design problems that help you to prepare final exam will also be released later today

- Leetcode problems
- Mostly about divide and conquer and dynamic programming

Final Exam

Final exam: May 3 (Friday) 8am-10am

- Location: TBH 180F
- Closed textbook exam
- You may use a sheet with notes on both sides, but not textbook and any other paper materials
- You may use a calculator, but not any device with transmitting functions, especially ones that can access the wireless or the Internet

Final Exam

True or False / Short answer

- Basic facts throughout the semester
- True or False: just answer yes/no, no justification needed
- Short answer: read the question carefully

Divide and Conquer

Dynamic programming

NP and NP-completeness

Final Exam

True or False / Short answer

- Basic facts throughout the semester
- True or False: just answer yes/no, no justification needed
- Short answer: read the question carefully

Di

Dy

N

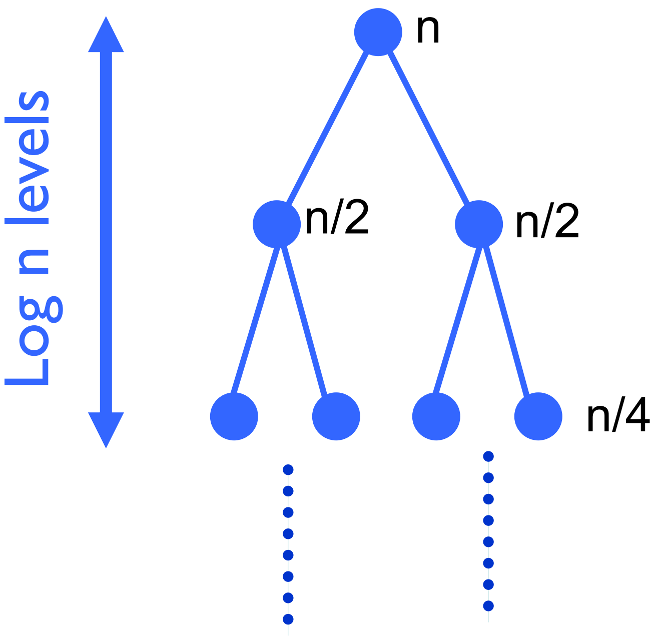
Cover knowledge learned throughout the semester!

Also check midterm review!

Divide and Conquer

Divide: We reduce a problem to several subproblems.

Typically, each sub-problem is at most a constant fraction of the size of the original problem



Conquer: Recursively solve each subproblem

Combine: Merge the solutions

Examples:

- Mergesort, Counting Inversions, Binary Search

Master Theorem

Suppose $T(n) = a\,T\left(\frac{n}{b}\right) + cn^k$ for all $n > b$. Then,

- If $a < b^k$ then $T(n) = \Theta\left(n^k\right)$
- If $a = b^k$ then $T(n) = \Theta\left(n^k \log n\right)$
- If $a > b^k$ then $T(n) = \Theta\left(n^{\log_b a}\right)$

Example: For mergesort algorithm we have

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n).$$

So, $k = 1, a = b^k$ and $T(n) = \Theta(n \log n)$

Problem 4 of Homework 3

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.

Divide and conquer: give an array, find the maximum subarray sum

- Divide: partition the array into two halves
- Conquer: Find the solution of each halves
- Combine: What if the solution contains elements from both halves? (for [6,-2, -3, 1,5], [6, -2] is from the first half, and [-3, 1, 5] is from the second half)

Problem 4 of Homework 3

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.

If the solution contains elements from both halves,

- The elements in the solution from the first half
  - Form an interval contains the last element of the first half
  - Have largest sum among all the intervals of the first half containing the last element of the first half ([6, -2] is the interval of [-2, -5, -6, -2] that (1) contains the last element and (2) has the largest sum)
- The elements in the solution from the second half
  - Form an interval contains the first element of the second half
  - Have largest sum among all the intervals of the second half containing the first element of the second half

Problem 4 of Homework 3

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.

Redefine problem: give an array, find

(1) s : the maximum subarray sum

(2) a : the maximum of subarray sum among all subarrays containing the first element

(3) b : the maximum of subarray sum among all subarrays containing the last element

Problem 4 of Homework 3

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.

Given s_1, a_1, b_1 for the first half, and s_2, a_2, b_2 for the second half, how to find the s, a, b?

- $s = \max\{s_1, s_2, b_1 + a_2\}$
- $a = \max\{a_1, sum_1 + a_2\}$
- $b = \max\{b_2, sum_2 + b_1\}$

sum_1: sum of all the elements in the first half

sum_2: sum of all the elements in the second half

T(n) = 2 T(n/2) + O(n) $\Longrightarrow$ T(n) = O(n log n)

Dynamic Programming

Principle:

- Optimal substructure: Remove certain part of the optimal solution (for the entire problem) is an optimal solution of a subproblem
- Carefully define a collection of subproblems. Typically, only a polynomial number of subproblems
- Parameterization/Memorization

Recipe:

- Find optimal substructure by investigating the optimal solution
- Find out additional assumptions/variables/subproblems that you need to do the induction
- Strengthen the hypothesis and define w.r.t. new subproblems

Dynamic programming techniques

- One dimensional dynamic programming: weighted interval scheduling, segmented least square
- Adding a new variable: knapsack.
- Order subproblems in the right way: RNA secondary structure / Shortest path with negative weights

P and NP

A decision problem is a computational problem where the answer is just yes/no

P: all decision problems solvable in polynomial time

NP: all decision problems with polynomial time certifier

- Certifier: Algorithm C(x, t) is a certifier for problem A if for every string x, the answer is “yes” iff there exists a string t such that C(x, t) = `yes`.

- Intuition: Certifier doesn't determine whether answer is “yes” on its own; rather, it checks a proposed proof t that answer is “yes”.

How to Prove a Problem is in NP?

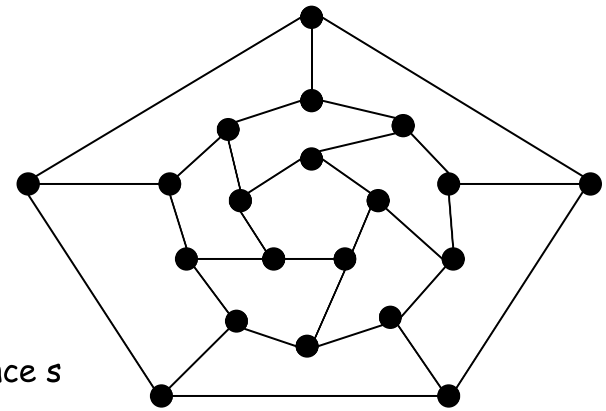
Make sure it is a decision problem, and give a polynomial time certifier

HAM-CYCLE. Given an undirected graph G = (V, E), does there exist a simple cycle C that visits every node?

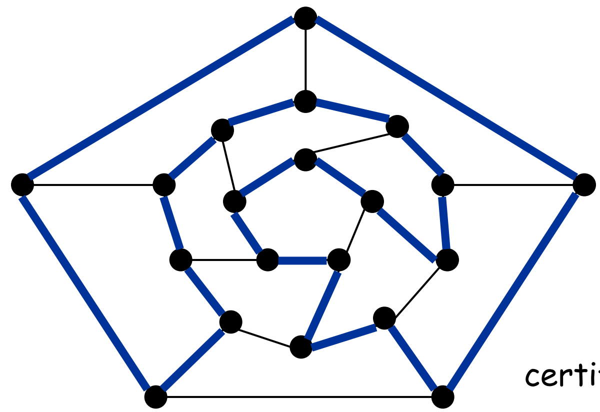
Certificate. A permutation of the n nodes.

Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.

Conclusion. HAM-CYCLE is in NP.



instance s



certificate t

NP Completeness

Hardest problems in NP

NP-complete: A problem Y in NP with the property that for every problem X in NP, $X \leq_p Y$.

Recipe to establish NP-completeness of problem Y.

- Step 1. Show that Y is in NP.
- Step 2. Choose an NP-complete problem X.
- Step 3. Prove that $X \leq_p Y$.

Def $A \leq_P B$: if there is an algorithm for problem A using a 'black box' (subroutine) that solve problem B s.t.,

- Algorithm uses only a polynomial number of steps
- Makes only a polynomial number of calls to a subroutine for **B**

Good Luck!