Goals:
- Prove polynomial time reduction from 3SAT to HamPath

Announcements:
- Go/middhacks2022
- Antiprocrastination Station go/ace

Questions
- Will the algorithm try to solve for all possible satisfying assignments or just one?
  - The 3SAT Solver should output YES if there is at least 1 satisfying assignment. It technically does not find the assignment.
- What is the non-polynomial runtime for each of 3SAT and HamPath?
  - Both can be solved in exponential time (test all possible assignments/all possible paths)
- Is there a way to reduce gadget graphs any more?
  - Maybe, but it might make the proof of correctness more difficult. Our goal is just polynomial time. We don't care if it is n^2 or n^10.
- Are there cases when making a visual proof is more functional than what we've done before with the boxes?
  - The box diagrams are not descriptive enough. But we can use pictures or pseudocode to describe the algorithm.
- Is there also a known way to solve Hamiltonian path using 3SAT?
  - Yes - 3SAT is NP-Hard
- Do we use graph gadgets once we get to more complex graphs?
  - This gadget is only used to turn 3SAT problems into HamPath problems. If you already have a HamPath problem of some complex graph, there is no need to convert.

2. Runtime: $|x| \approx n^3 \Rightarrow |x|^{1/3} = n$

$|x| = (\text{size of 3SAT input}) = O(\#\text{ of clauses}) = O(n^3)$ (# of clauses $= m$)

n variables; ways of choosing 3 variables to put in clause: $\binom{n}{3} \cdot 8 = O(n^3)$ (8: choose where to put negations)

¬ ¬ ¬
¬ ¬ 7
¬ 7 ¬

$2 \times 2 \times 2 = 8$

$G(x)$: # of vertices: $n \cdot O(m) + O(m) = O(nm)$ (gadgets; cross diamond vertices; extra clause vertices)

# of edges: $O(n^2 m^2) = O(n^2 (n^3)^2) = O(n^8)$

$n = |x|^{1/3}$

Runtime: $O(|x|^{8/3})$

Polynomial!

3. a) If HamPath$(G(x))$ = Yes, then 3SAT(x) = Yes
b) If 3SAT(x) = Yes, HamPath$(G(x))$ = Yes

a) If HamPath$(G(x))$ Yes,
- → We argued each gadget must have LRL or RLR path
- → The path must hit each c_i vertex exactly 1.
  - It must hit c_i from a gadget u_j and return to gadget u_j because it would skip vertices, or repeating vertices
  - The variable u_j causes c_i to be satisfied

get a satisfying assignment → 3SAT(x) is Yes
where LRL → True RLR → False

b) If 3SAT(x) = Yes, then there is a satisfying assignment
- → each u_i gets assigned T or F
- → clause j has at least one satisfying variables.

If u_i is TRUE → LRL through i^{th} gadget
If u_i is False → RLR "
Pick a variable that satisfies clause j, and jog out to c_j from that gadget.

∴ This gives us a path that hits all vertices exactly once in $G(x)$. ☐ QED

Once you prove Π is NP-Hard, can use in place of 3-SAT:

Theorem: • If Π (NP-Hard) $\le_p$ Q, then Q ∈ NP-Hard:

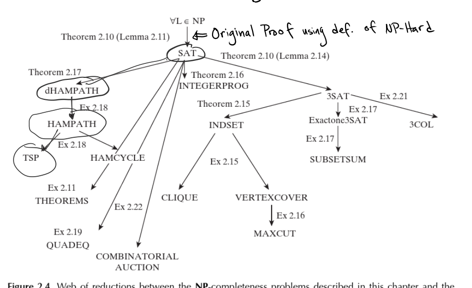

Figure 2.4. Web of reductions between the **NP**-completeness problems described in this chapter and the exercises. Thousands more are known.

(From Arora + Barak)

13s: Dijkstra's Algorithm

Saturday, May 8, 2021
12:15 PM

Shortest Path Problem

Input: • $G = (V, E)$, $|V| = n$, $|E| = m$
• $s, t \in V$

Output: Shortest path from s to t

* with appropriate adjacency list

Conditions	Algorithm	Runtime*
negative edge weights	Bellman-Ford	$O(nm)$
no negative edges weights	Dijkstra's	$O((n+m)\log n)$
all edges have weight 1	Breadth First Search	$O(n+m)$