18.404 Recitation 7

Oct 23, 2020

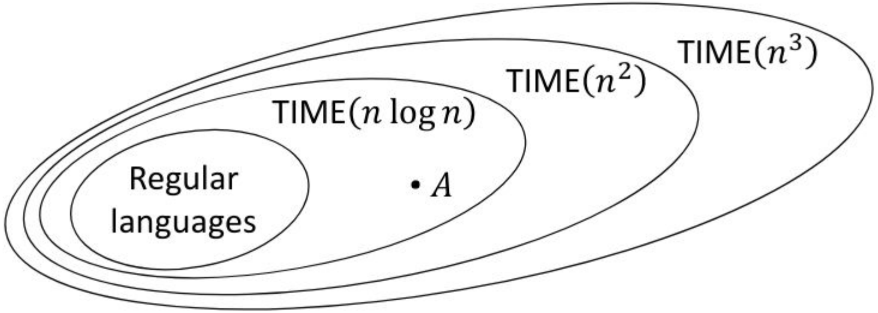
Today's Topics

- Formal Definition: P, NP
- Review: PATH
  - PATH ∈ P
- Review: HAMPATH
  - HAMPATH ∈ NP
- Polynomial Time Reducibility
- Example Reductions — Proving NP-Complete
  - 3-SAT $\leq_p$ SUBSET-SUM
  - HAMPATH $\leq_p$ UHAMPATH

Formal Definition: P

TIME

- Let t: $\mathbb{N} \rightarrow \mathbb{N}$
- Say M runs in time t if TM M always halts within $t(n)$ steps on all inputs of length n



Formal Definition: P (cont.)

TIME(n^k) = { B | some 1-tape deterministic TM decides B in O(t(n^k)) steps)

P = $\cup_k$ TIME(n^k)

= polynomial time decidable languages

Corresponds roughly to realistically solvable problems

Formal Definition: NP

NTIME

- Let $t: \mathbb{N} \rightarrow \mathbb{N}$
- A NTM M runs in time $t(n)$ if all branches halt within $t(n)$ steps on all inputs of length n
- NTIME($t(n)$) = { B | some 1-tape NTM decides B and runs in O($t(n)$) steps }

Formal Definition: NP (cont.)

$NP = \bigcup_k NTIME(n^k)$

$\quad$ = nondeterministic polynomial time decidable languages

Corresponds roughly to easily verifiable problems

Intuitions: P vs. NP

P - All languages where one can test membership quickly

- Problem presented to nondet. TM solvable in polynomial time

NP - All languages where one can verify membership quickly

- Problem + solution presented to nondet. TM verified in polynomial time

$P \subseteq NP$, but unknown whether $P = NP$ or $P \neq NP$

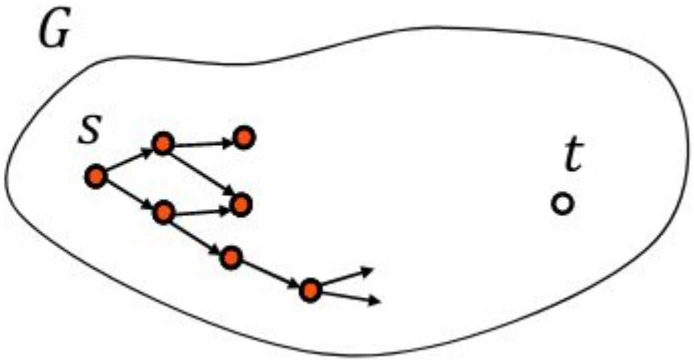
Review: PATH

PATH = {<G, s, t> | G is a directed graph with path from s to t }

Thm: PATH ∈ P

Proof: M = “On input <G, s, t>

1. Run BFS on G starting at s
2. Accept if t is reached.
   Reject otherwise”



Review: HAMPATH

HAMPATH = {<G, s, t> | G is a directed graph with path from s to t
and path goes through every node of G without repeats }

Thm: HAMPATH $\in$ NP

Proof: M = “On input <G, s, t> (G has m nodes)

1. Nondeterministically pick sequence $v_1, v_2, \ldots, v_m$ of m nodes
2. Accept if $v_1 = s$, $v_m = t$
   each (v_i, v_{i+1}) is an edge and v_i does not repeat
3. Reject if any condition fails”

Polynomial Time Reducibility

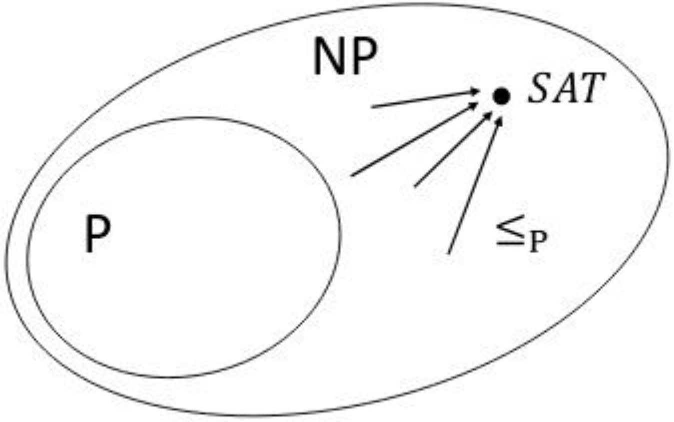
Definition: A is polynomial time reducible to B ($A \le_p B$) if $A \le_m B$ by a reduction function *m* that is computable in polynomial time

Thm: If $A \le_p B$ and $B \in P$, then $A \in P$

f is computable in polynomial time

Polynomial Time Reducibility (cont.)

Corollary: If SAT ∈ P, then P = NP



Idea to show $SAT \in P \rightarrow P = NP$

Define: SUBSET-SUM

Language definition

Given a collection of numbers $x_1, \ldots, x_k$ and a target number t

Does the collection contain a subcollection of numbers which sum up to t?

ex) {1, 2, 3, 5, 7}, t=13 $\in$ SUBSET-SUM

$$t = 20 \notin \text{SUBSET-SUM}$$

Example Reduction: 3-SAT $\leq_p$ SUBSET-SUM

Proving SUBSET-SUM is NP-Complete

1. SUBSET-SUM $\in$ NP
2. NP-Complete Language (3-SAT) $\leq_p$ SUBSET-SUM

Simpler Question First: Is SUBSET-SUM $\in$ NP?

Example Reduction: 3-SAT $\leq_p$ SUBSET-SUM (cont.)

Proving SUBSET-SUM is NP-Complete: 3-SAT $\leq_p$ SUBSET-SUM

Idea:

- Find way to convert *any* 3-SAT problem to a SUBSET-SUM problem
- SUBSET-SUM problem should somehow simulate solving 3-SAT formula
- Make sure conversion is polynomial in time!

Therefore, solving SUBSET-SUM problem, basically also solving 3-SAT problem

Example Reduction: 3-SAT $\leqslant_p$ SUBSET-SUM (cont.)

Construction: Assume l variable $x_1 \ldots x_l$, assume k clauses $c_1 \ldots c_k$

For every variable x_i produce two digits y_i, z_i (for SUBSET-SUM problem)

For every clause c_j produce two digits g_j, h_j (for SUBSET-SUM problem)

Example Reduction: 3-SAT $\leqslant_p$ SUBSET-SUM (cont.)

Construction: $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\overline{x_3} \vee \cdots \vee \cdots)$

Represents variables

Represents clauses

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
h_1							1	0	$\cdots$	0
g_2								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

Define: UHAMPATH

Recall:

HAMPATH = {<G, s, t> | G is a directed graph with path from s to t
and path goes through every node of G without repeats }

UHAMPATH = {<G, s, t> | G is a undirected graph with path from s to t
and path goes through every node of G without repeats }

Example Reduction: HAMPATH $\leq_p$ UHAMPATH cont.

Proving UHAMPATH is NP-Complete

1. UHAMPATH $\in$ NP
2. NP-Complete Language (HAMPATH) $\leq_p$ UHAMPATH

Simpler Question First: Is UHAMPATH $\in$ NP?

Example Reduction: HAMPATH $\leq_p$ UHAMPATH cont.

Proving UHAMPATH is NP-Complete: HAMPATH $\leq_p$ UHAMPATH

Idea:

- Convert HAMPATH directed graph G to UHAMPATH undirected graph G' where:
  - <G,s,t> ∈ HAMPATH *iff* <G', s', t'> ∈ UHAMPATH
  - <G,s,t> ∉ HAMPATH *iff* <G', s', t'> ∉ UHAMPATH
- Make sure conversion is polynomial in time!

Example Reduction: HAMPATH $\leq_p$ UHAMPATH cont.

Construction: Convert every node *u* in HAMPATH G, to three nodes in G'

- u → u^{in}, u^{mid}, u^{out}
- s → s^{out}
- t → t^{in}

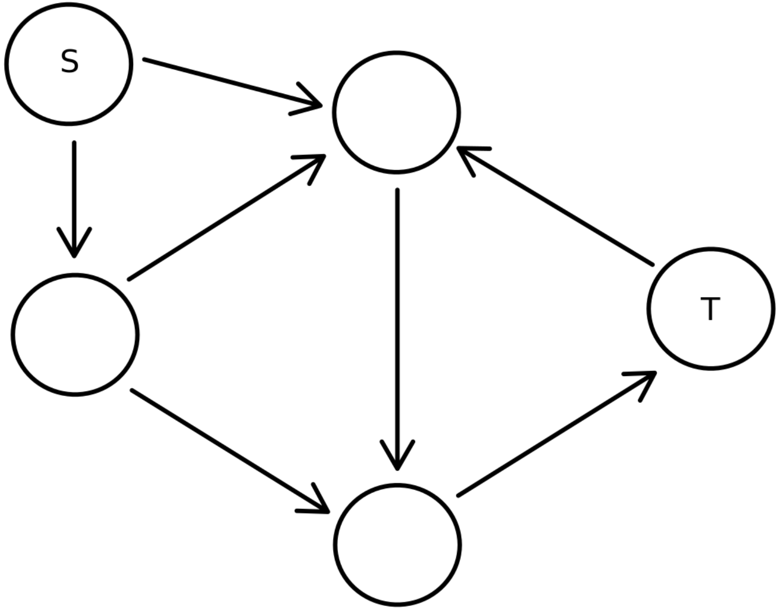
HAMPATH path: $s, u_1, u_2, \ldots, u_k, t,$

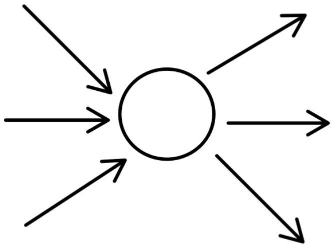
gets converted to

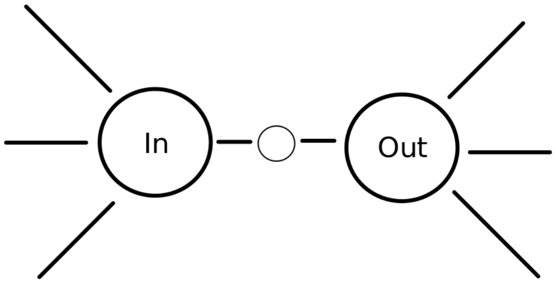
UHAMPATH path: $s^{\text{out}},\ u_1^{\text{in}}, u_1^{\text{mid}}, u_1^{\text{out}},\ u_2^{\text{in}}, u_2^{\text{mid}}, u_2^{\text{out}}, \ldots, t^{\text{in}}$

Example Reduction: HAMPATH $\leqslant_p$ UHAMPATH cont.

Construction by example:







Example Reduction: HAMPATH $\leqslant_p$ UHAMPATH cont.

Construction by example: