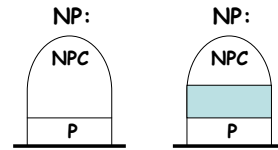


CS151 Complexity Theory

Lecture 4
April 5, 2007

Ladner's Theorem

- Assuming $P \neq NP$, what does the world (inside NP) look like?



April 5, 2007

CS151 Lecture 4

2

Ladner's Theorem

Theorem (Ladner): If $P \neq NP$, then there exists $L \in NP$ that is neither in P nor NP -complete.

- Proof:
 - can enumerate (TMs M_i deciding) all languages in P
 - can enumerate (TMs N_i deciding) all NP -complete languages

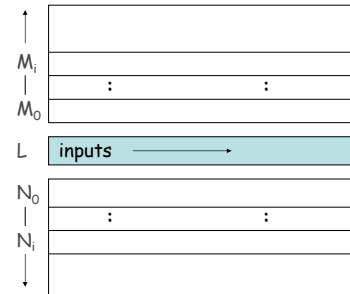
April 5, 2007

CS151 Lecture 4

3

Ladner's Theorem

- Our goal: $L \in NP$ that is neither in P nor NP -complete



April 5, 2007

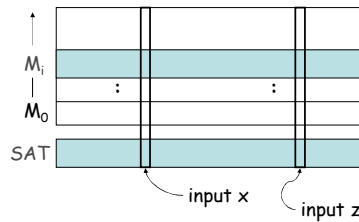
CS151 Lecture 4

4

Ladner's Theorem

- Top half, assuming $P \neq NP$:

- focus on M_i
- for any x , can always find some $z \geq x$ on which M_i and SAT differ (why?)



April 5, 2007

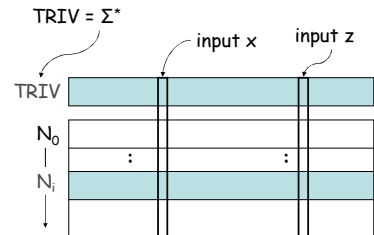
CS151 Lecture 4

5

Ladner's Theorem

- Bottom half, assuming $P \neq NP$:

- focus on N_i
- for any x , can always find some $z \geq x$ on which N_i and TRIV differ (why?)



April 5, 2007

CS151 Lecture 4

6

Ladner's Theorem

- Try to "merge":

SAT

TRIV

- on input x , either
 - answer SAT(x)
 - answer TRIV(x)
- if can decide which one in P , $L \in NP$

April 5, 2007 CS151 Lecture 4 7

Ladner's Theorem

- General scheme: $f(n)$ slowly increasing function

L

...

SAT

TRIV

$f(|x|)$

0	0	0	1	1	1	2	2	2	2	...
---	---	---	---	---	---	---	---	---	---	-----

- $f(|x|)$ even: answer SAT(x)
- $f(|x|)$ odd: answer TRIV(x)

- notice choice only depends on length of input... that's OK

April 5, 2007 CS151 Lecture 4 8

Ladner's Theorem

- 1st attempt to define $f(n)$
- "eager $f(n)$ ": increase at 1st opportunity
- Inductive definition: $f(0) = 0$; $f(n) =$
 - if $f(n-1) = 2i$, trying to kill M_i
 - if $\exists z < 1^n$ s.t. $M_i(z) \neq \text{SAT}(z)$, then $f(n) = f(n-1) + 1$; else $f(n) = f(n-1)$
 - if $f(n-1) = 2i+1$, trying to kill N_i
 - if $\exists z < 1^n$ s.t. $N_i(z) \neq \text{TRIV}(z)$, then $f(n) = f(n-1) + 1$; else $f(n) = f(n-1)$

April 5, 2007 CS151 Lecture 4 9

Ladner's Theorem

- Problem: eager $f(n)$ too difficult to compute
- on input of length n ,
 - look at all strings z of length $< n$
 - compute SAT(z) or $N_i(z)$ for each !
- Solution: "lazy" $f(n)$
 - on input of length n , only run for $2n$ steps
 - if enough time to see should increase (over $f(n-1)$), do it; else, stay same
 - (alternate proof: give explicit $f(n)$ that grows slowly enough...)

April 5, 2007 CS151 Lecture 4 10

Ladner's Theorem

- Key: n eventually large enough to notice completed previous stage

M_i

L

...

$f(|x|)$

0	0	1	1	...	k	k	k	k	k	...
---	---	---	---	-----	---	---	---	---	---	-----

input $z < x$ input x

suppose $k = 2i$

• I'm supposed to ensure M_i is killed

• I finally have enough time to check input z

• I notice z did the job, increase f to $k+1$

April 5, 2007 CS151 Lecture 4 11

Ladner's Theorem

- Inductive definition of $f(n)$
 - $f(0) = 0$
 - $f(n)$: for n steps compute $f(0), f(1), f(2), \dots$

L

...

f

0	0	1	1	...	k	k	k	...
---	---	---	---	-----	---	---	---	-----

got this far in n steps input x , $|x| = n$

April 5, 2007 CS151 Lecture 4 12

Ladner's Theorem

- if $k = 2i$:
 - for n steps try (lex order) to find z s.t. $SAT(z) \neq M_i(z)$ and $f(|z|)$ even
 - if found, $f(n) = f(n-1)+1$ else $f(n-1)$
- if $k = 2i + 1$:
 - for n steps try (lex order) to find z s.t. $TRIV(z) \neq N_i(z)$ and $f(|z|)$ odd
 - if found, $f(n) = f(n-1)+1$ else $f(n-1)$

April 5, 2007

CS151 Lecture 4

13

Ladner's Theorem

- Finishing up:
 - $L = \{ x \mid x \in SAT \text{ if } f(|x|) \text{ even,}$
 $x \in TRIV \text{ if } f(|x|) \text{ odd} \}$
- $L \in \mathbf{NP}$ since $f(|x|)$ can be computed in $O(n)$ time

April 5, 2007

CS151 Lecture 4

14

Ladner's Theorem

- suppose M_i decides L
 - f gets stuck at $2i$
 - $L \equiv SAT$ for $z : |z| > n_0$
 - implies $SAT \in \mathbf{P}$. Contradiction.
- suppose N_i decides L
 - f gets stuck at $2i+1$
 - $L \equiv TRIV$ for $z : |z| > n_0$
 - implies $L(N_i) \in \mathbf{P}$. Contradiction.
- (last of diagonalization...)

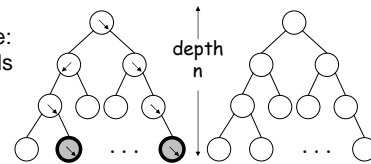
April 5, 2007

CS151 Lecture 4

15

A puzzle

A puzzle:
two kinds
of trees



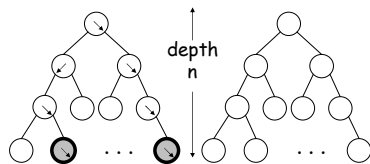
- cover up nodes with c colors
- promise: never color "arrow" same as "blank"
- determine which kind of tree in $\text{poly}(n, c)$ steps?

April 5, 2007

CS151 Lecture 4

16

A puzzle

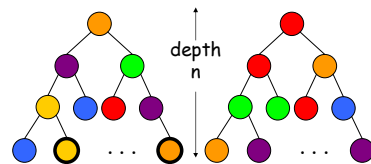


April 5, 2007

CS151 Lecture 4

17

A puzzle



April 5, 2007

CS151 Lecture 4

18

Introduction

- Ideas
 - depth-first-search; stop if see \odot
 - how many times may we see a given “arrow color”?
 - at most $n+1$
 - pruning rule?
 - if see a color $> n+1$ times, it can't be an arrow node; prune
 - # nodes visited before know answer?
 - at most $c(n+2)$

April 5, 2007

CS151 Lecture 4

19

Sparse languages and NP

- We often say **NP**-complete languages are “hard”
- More accurate: **NP**-complete languages are “expressive”
 - lots of languages reduce to them

April 5, 2007

CS151 Lecture 4

20

Sparse languages and NP

- Sparse language: one that contains at most $\text{poly}(n)$ strings of length $\leq n$
- not very expressive – can we show this cannot be **NP**-complete (assuming $\mathbf{P} \neq \mathbf{NP}$) ?
 - yes: Mahaney '82 (homework problem)
- Unary language: subset of 1^* (at most n strings of length $\leq n$)

April 5, 2007

CS151 Lecture 4

21

Sparse languages and NP

Theorem (Berman '78): if a unary language is **NP**-complete then $\mathbf{P} = \mathbf{NP}$.

- Proof:
 - let $U \subset 1^*$ be a unary language and assume $\text{SAT} \leq U$ via reduction R
 - $\varphi(x_1, x_2, \dots, x_n)$ instance of SAT

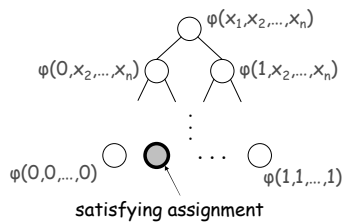
April 5, 2007

CS151 Lecture 4

22

Sparse languages and NP

– self-reduction tree for φ :



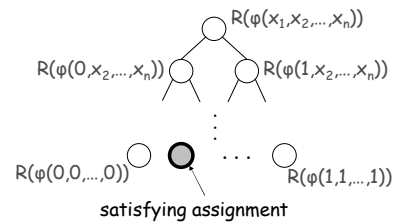
April 5, 2007

CS151 Lecture 4

23

Sparse languages and NP

– applying reduction R :



April 5, 2007

CS151 Lecture 4

24

Sparse languages and NP

- on input of length $m = |\varphi(x_1, x_2, \dots, x_n)|$, R produces string of length $\leq p(m)$
- R's different outputs are "colors"
 - 1 color for strings not in 1^*
 - at most $p(m)$ other colors
- puzzle solution \Rightarrow can solve SAT in $\text{poly}(p(m)+1, n+1) = \text{poly}(m)$ time!

April 5, 2007

CS151 Lecture 4

25

Summary

- nondeterministic time classes:
NP, coNP, NEXP
- NTIME Hierarchy Theorem:
NP \neq NEXP
- major open questions:

$$P \stackrel{?}{=} NP$$

$$NP \stackrel{?}{=} \text{coNP}$$

April 5, 2007

CS151 Lecture 4

26

Summary

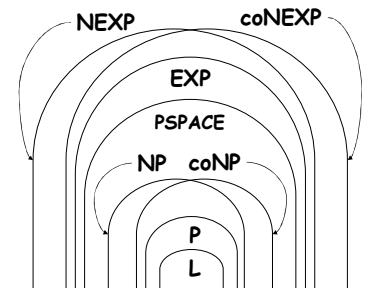
- **NP**-"intermediate" problems (unless **P = NP**)
 - technique: delayed diagonalization
- unary languages not **NP**-complete (unless **P = NP**)
 - true for sparse languages as well (homework)
- complete problems:
 - circuit SAT is **NP**-complete
 - UNSAT is **coNP**-complete
 - succinct circuit SAT is **NEXP**-complete

April 5, 2007

CS151 Lecture 4

27

Summary



April 5, 2007

CS151 Lecture 4

28

Remainder of lecture

- nondeterminism applied to space
- reachability
- two surprises:
 - Savitch's Theorem
 - Immerman/Szelepcsényi Theorem

April 5, 2007

CS151 Lecture 4

29

Nondeterministic space

- **NSPACE(f(n))** = languages decidable by a multi-tape NTM that touches at most $f(n)$ squares of its work tapes *along any computation path*, where n is the input length, and $f: \mathbf{N} \rightarrow \mathbf{N}$

April 5, 2007

CS151 Lecture 4

30

Nondeterministic space

- Robust nondeterministic space classes:

$$NL = NSPACE(\log n)$$

$$NPSPACE = \cup_k NSPACE(n^k)$$

April 5, 2007

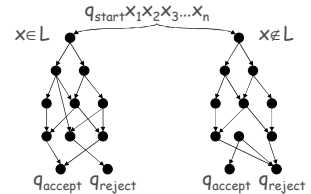
CS151 Lecture 4

31

Reachability

- Recall: at most n^k configurations of given NTM M running in **NSPACE**($\log n$).

- easy to determine if C yields C' in one step
- configuration graph for M on input x :



April 5, 2007

CS151 Lecture 4

32

Reachability

- Conclude: $NL \subset P$
– and $NPSPACE \subset EXP$
- S-T-Connectivity (STCONN): given directed graph $G = (V, E)$ and nodes s, t , is there a path from s to t ?

Theorem: STCONN is **NL**-complete under logspace reductions.

April 5, 2007

CS151 Lecture 4

33

Reachability

- Proof:
 - in **NL**: guess path from s to t one node at a time
 - given $L \in \mathbf{NL}$ decided by NTM M construct configuration graph for M on input x (can be done in logspace)
 - s = starting configuration; $t = q_{\text{accept}}$

April 5, 2007

CS151 Lecture 4

34

Two startling theorems

- Strongly believe $P \neq NP$
- nondeterminism seems to add enormous power
- for space: Savitch '70:

$$NPSPACE = PSPACE$$

and

$$NL \subset SPACE(\log^2 n)$$

April 5, 2007

CS151 Lecture 4

35

Two startling theorems

- Strongly believe $NP \neq \text{coNP}$
- seems impossible to convert existential into universal
- for space: Immerman/Szelepcényi '87/'88:

$$NL = \text{coNL}$$

April 5, 2007

CS151 Lecture 4

36

Savitch's Theorem

Theorem: STCONN \in SPACE($\log^2 n$)

- Corollary: NL \subset SPACE($\log^2 n$)
- Corollary: NPSPACE = PSPACE

April 5, 2007

CS151 Lecture 4

37

Proof of Theorem

- input: $G = (V, E)$, two nodes s and t
- recursive algorithm:

```

/* return true iff path from x to y of length at most 2^i */
PATH(x, y, i)
if i = 0 return ( x = y or (x, y)  $\in$  E )      /* base case */
for z in V
  if PATH(x, z, i-1) and PATH(z, y, i-1) return(true);
return(false);
end
    
```

April 5, 2007

CS151 Lecture 4

38

Proof of Theorem

- answer to STCONN: PATH($s, t, \log n$)
- space used:
 - (depth of recursion) \times (size of "stack record")
- depth = $\log n$
- claim stack record: "(x, y, i)" sufficient
 - size $O(\log n)$
- when return from PATH(a, b, i) can figure out what to do next from record (a, b, i) and previous record

April 5, 2007

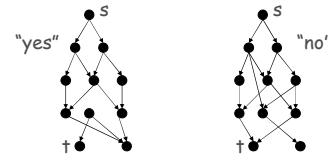
CS151 Lecture 4

39

I-S Theorem

Theorem: ST-NON-CONN \in NL

- Proof: slightly tricky setup:
 - input: $G = (V, E)$, two nodes s, t



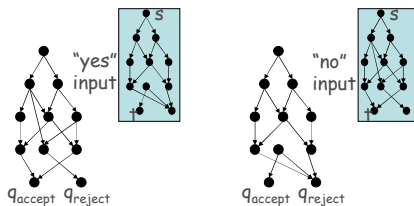
April 5, 2007

CS151 Lecture 4

40

I-S Theorem

- want nondeterministic procedure using only $O(\log n)$ space with behavior:



April 5, 2007

CS151 Lecture 4

41

I-S Theorem

- observation: given **count** of # nodes reachable from s , can solve problem
 - for each $v \in V$, *guess* if it is reachable
 - if yes, *guess* path from s to v
 - if guess doesn't lead to v , reject.
 - if $v = t$, reject.
 - else counter++
 - if counter = **count** accept

April 5, 2007

CS151 Lecture 4

42

I-S Theorem

- every computation path has sequence of guesses...
- only way computation path can lead to accept:
 - correctly guessed reachable/unreachable for each node v
 - correctly guessed path from s to v for each reachable node v
 - saw *all* reachable nodes
 - t not among reachable nodes

April 5, 2007

CS151 Lecture 4

43

I-S Theorem

- $R(i)$ = # nodes reachable from s in at most i steps
- $R(0) = 1$: node s
- we will compute $R(i+1)$ from $R(i)$ using $O(\log n)$ space and nondeterminism
- computation paths with “bad guesses” all lead to reject

April 5, 2007

CS151 Lecture 4

44

I-S Theorem

- Outline: in n phases, compute $R(1), R(2), R(3), \dots, R(n)$
- only $O(\log n)$ bits of storage between phases
- in end, lots of computation paths that lead to reject
- only computation paths that survive have computed correct value of $R(n)$
- apply observation.

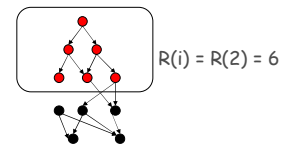
April 5, 2007

CS151 Lecture 4

45

I-S Theorem

- computing $R(i+1)$ from $R(i)$:



- Initialize $R(i+1) = 0$
- For each $v \in V$, *guess* if v reachable from s in at most $i+1$ steps

April 5, 2007

CS151 Lecture 4

46

I-S Theorem

- if “yes”, *guess* path from s to v of at most $i+1$ steps. Increment $R(i+1)$
- if “no”, visit $R(i)$ nodes reachable in at most i steps, check that none is v or adjacent to v
 - for $u \in V$ *guess* if reachable in $\leq i$ steps; *guess* path to u ; counter++
 - KEY: if counter $\neq R(i)$, reject
 - at this point: **can be sure v not reachable**

April 5, 2007

CS151 Lecture 4

47

I-S Theorem

- correctness of procedure:
 - two types of errors we can make
 - (1) might guess v is reachable in at most $i+1$ steps when it is not
 - won't be able to guess path from s to v of correct length, so we will reject.
- “easy” type of error

April 5, 2007

CS151 Lecture 4

48

I-S Theorem

- (2) might guess v is **not** reachable in at most $i+1$ steps when it is
 - then must **not** see v or neighbor of v while visiting nodes reachable in i steps.
 - but forced to visit $R(i)$ distinct nodes
 - therefore must try to visit node v that is **not** reachable in $\leq i$ steps
 - won't be able to guess path from s to v of correct length, so we will reject.

"easy" type of error

April 5, 2007

CS151 Lecture 4

49

Summary

- nondeterministic space classes
NL and **NPSPACE**
- ST-CONN **NL**-complete

April 5, 2007

CS151 Lecture 4

50

Summary

- Savitch: **NPSPACE = PSPACE**
 - Proof: ST-CONN \in **SPACE($\log^2 n$)**
 - open question:
NL = L?
- Immerman/Szelepcsényi : **NL = coNL**
 - Proof: ST-NON-CONN \in **NL**

April 5, 2007

CS151 Lecture 4

51