A puzzle

• cover up nodes with c colors
• promise: never color “arrow” same as “blank”
• determine which kind of tree in poly(n, c) steps?

A puzzle

depth n

A puzzle

depth n

Introduction

• Ideas
  – depth-first-search; stop if see
  – 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)

Sparse languages and NP

• We often say NP-compete languages are “hard”

• More accurate: NP-complete languages are “expressive”
  – lots of languages reduce to them
Sparse languages and NP

- Sparse language: one that contains at most poly(n) strings of length ≤ n
- not very expressive – can we show this cannot be NP-complete (assuming P ≠ NP)?
  - yes: Mahaney ’82 (homework problem)

- Unary language: subset of 1* (at most n strings of length ≤ n)

Theorem (Berman ’78): if a unary language is NP-complete then P = NP.

- Proof:
  - let U ⊆ 1* be a unary language and assume SAT ≤ U via reduction R
  - φ(x₁,x₂,…,xₙ) instance of SAT

Sparse languages and NP

- self-reduction tree for φ:

Sparse languages and NP

- applying reduction R:

Sparse languages and NP

- on input of length m = |φ(x₁,x₂,…,xₙ)|, R produces string of length ≤ p(m)
- R’s different outputs are “colors”
  - 1 color for strings not in 1*
  - at most p(m) other colors
- puzzle solution ⇒ can solve SAT in poly(p(m)+1, n+1) = poly(m) time!

Summary

- nondeterministic time classes: 
  NP, coNP, NEXP
- NTIME Hierarchy Theorem:
  NP ≠ NEXP
- major open questions:
  P ≠ NP
  NP ≠ coNP
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

Remainder of lecture

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

Nondeterministic space

- Robust nondeterministic space classes:
  \[
  NL = \text{NSPACE}(\log n) \\
  \text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k)
  \]
Reachability

• Conclude: \( \text{NL} \subseteq \text{P} \)
  – and \( \text{NPSPACE} \subseteq \text{EXP} \)

• 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.

Two startling theorems

• Strongly believe \( \text{P} \neq \text{NP} \)
• nondeterminism seems to add enormous
  power
• for space: Savitch ‘70:
  \[ \text{NPSPACE} = \text{PSPACE} \]
  and
  \[ \text{NL} \subseteq \text{SPACE}(\log^2 n) \]

Savitch’s Theorem

**Theorem:** STCONN \( \in \text{SPACE}(\log^2 n) \)

• Corollary: \( \text{NL} \subseteq \text{SPACE}(\log^2 n) \)
• Corollary: \( \text{NPSPACE} = \text{PSPACE} \)

Two startling theorems

• Strongly believe \( \text{NP} \neq \text{coNP} \)
• seems impossible to convert existential
  into universal
• for space: Immerman/Szelepscényi ‘87/’88:
  \[ \text{NL} = \text{coNL} \]

Proof of Theorem

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

```plaintext
/* 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);
```

Reachability

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

– answer to STCONN: PATH(s, t, log n)
– space used:
  • (depth of recursion) x (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

Nondeterministic space

• Robust nondeterministic space classes:
  \[ \text{NL} = \text{NSPACE}(\log n) \]
  \[ \text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k) \]

Second startling theorem

• Strongly believe \( \text{NP} \neq \text{coNP} \)
• seems impossible to convert existential into universal
• for space: Immerman/Szelepcsényi '87/'88:
  \[ \text{NL} = \text{coNL} \]

I-S Theorem

Theorem: ST-NON-CONN \( \in \text{NL} \)

• Proof: slightly tricky setup:
  – input: \( G = (V, E) \), two nodes s, t

I-S Theorem

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

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
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 8, 2015

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 8, 2015

I-S Theorem

-- Outline: in \( n \) phases, compute \( R(1), R(2), R(3), \ldots R(n) \)
-- 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 8, 2015

I-S Theorem

-- computing \( R(i+1) \) from \( R(i) \):
  \[ R(i) = R(2) = 6 \]

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

  \[ R(i) = R(2) = 6 \]

-- Outline: in \( n \) phases, compute \( R(1), R(2), R(3), \ldots R(n) \)
-- 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 8, 2015

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 ≠ \( R(i) \), reject
  • at this point: can be sure \( v \) not reachable

April 8, 2015

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 8, 2015
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 ≤ 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 8, 2015

Summary

• nondeterministic space classes
  \( \text{NL} \) and \( \text{NPSPACE} \)

• \( \text{ST-CONN} \) \( \text{NL} \)-complete

April 8, 2015

Introduction

Power from an unexpected source?

• we know \( \text{P} \neq \text{EXP} \), which implies no poly-time \text{algorithm} for Succinct CVAL
• poly-size Boolean \text{circuits} for Succinct CVAL ??

Does \( \text{NP} \) have linear-size, log-depth Boolean circuits ??

April 8, 2015

Outline

• Boolean circuits and formulas
• uniformity and advice
• the \text{NC} hierarchy and parallel computation
• the quest for circuit lower bounds
• a lower bound for formulas

April 8, 2015

Summary

• Savitch: \( \text{NPSPACE} = \text{PSPACE} \)
  – Proof: \( \text{ST-CONN} \in \text{SPACE}(\log^2 n) \)
  – open question:
    \[ \text{NL} = \text{L} ? \]

• Immerman/Szelepcsényi : \( \text{NL} = \text{coNL} \)
  – Proof: \( \text{ST-NON-CONN} \in \text{NL} \)

April 8, 2015

Boolean circuits

• circuit \( C \)
  – directed acyclic graph
  – nodes: AND (\( \wedge \)); OR (\( \vee \)); NOT (\( \neg \)); variables \( x_i \)
  \[ x_1, x_2, x_3, \ldots, x_n \]

• \( C \) computes function \( f: \{0,1\}^n \rightarrow \{0,1\} \) in natural way
  – identify \( C \) with function \( f \) it computes

April 8, 2015
**Boolean circuits**

- **size** = # gates
- **depth** = longest path from input to output
- **formula (or expression)**: graph is a tree

- every function $f: \{0,1\}^n \rightarrow \{0,1\}$ computable by a circuit of size at most $O(n^{2^n})$
  - AND of $n$ literals for each $x$ such that $f(x) = 1$
  - OR of up to $2^n$ such terms

**Circuit families**

- circuit works for specific input length
- we’re used to $f: \sum \rightarrow \{0,1\}$
- circuit family : a circuit for each input length $C_1, C_2, C_3, \ldots = \{C_n\}$
- “$\{C_n\}$ computes $f$” iff for all $x$
  - $C_n(x) = f(x)$
- “$\{C_n\}$ decides $L$”, where $L$ is the language associated with $f$

**Connection to TMs**

- given TM $M$ running in time $t(n)$ decides language $L$
- can build circuit family $\{C_n\}$ that decides $L$
  - size of $C_n = O(t(n)^2)$
  - Proof: CVAL construction
- Conclude: $L \in \text{P}$ implies family of polynomial-size circuits that decides $L$

**Uniformity**

- Strange aspect of circuit family:
  - can “encode” (potentially uncomputable) information in family specification
- solution: **uniformity** – require specification is simple to compute
  - **Definition:** circuit family $\{C_n\}$ is logspace uniform iff TM $M$ outputs $C_n$ on input $1^n$ and runs in $O(\log n)$ space

**Theorem:** $\text{P} = \text{languages decidable by logspace uniform, polynomial-size circuit families } \{C_n\}$

- Proof:
  - already saw $(\Rightarrow)$
  - $(\Leftarrow)$ on input $x$, generate $C_{|x|}$ evaluate it and accept iff output = 1
TM's that take advice

- Family \( \{C_n\} \) without uniformity constraint is called "non-uniform"
- Regard "non-uniformity" as a limited resource just like time, space, as follows:
  - Add read-only "advice" tape to TM \( M \)
  - \( M \) "decides \( L \) with advice \( A(n) \)" iff
    \[ M(x, A(|x|)) \text{ accepts } x \in L \]
  - Note: \( A(n) \) depends only on \( |x| \)

TM's that take advice

Definition: \( \text{TIME}(t(n))/f(n) = \) the set of those languages \( L \) for which:
- There exists \( A(n) \) s.t. \( |A(n)| \leq f(n) \)
- TM \( M \) decides \( L \) with advice \( A(n) \) in time \( t(n) \)
- Most important such class:
  \[ \text{P/poly} = \bigcup_k \text{TIME}(n^k)/n^k \]

TM's that take advice

**Theorem**: \( L \in \text{P/poly} \) iff \( L \) decided by family of (non-uniform) polynomial size circuits.

- Proof:
  - \((\Rightarrow)\) \( C_n \) from CVAL construction; hardwire advice \( A(n) \)
  - \((\Leftarrow)\) Define \( A(n) = \) description of \( C_n \); on input \( x \), TM simulates \( C_{|x|}(x) \)

Approach to P/NP

- Believe \( \text{NP} \not\subset \text{P} \)
  - Equivalent: "\( \text{NP} \) does not have uniform, polynomial-size circuits"
- **Even believe \( \text{NP} \not\subset \text{P/poly} \)**
  - Equivalent: "\( \text{NP} \) (or, e.g. SAT) does not have polynomial-size circuits"
  - Imply \( \text{P} \neq \text{NP} \)
  - Many believe: best hope for \( \text{P} \neq \text{NP} \)