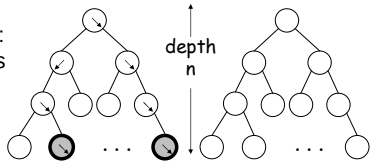
CS151
Complexity Theory

Lecture 4

April 8, 2004

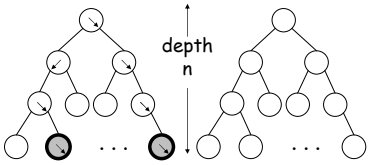
Introduction

A puzzle: two kinds of trees

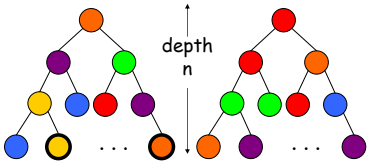


- cover up nodes with c colors
- promise: never color "arrow" same as "blank"
- determine which kind of tree in poly(n, c) steps?

A puzzle



A puzzle



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)

Outline

- sparse languages and **NP**
- nondeterminism applied to space
- reachability
- Savitch's Theorem
- Immerman/Szelepcsényi Theorem

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)

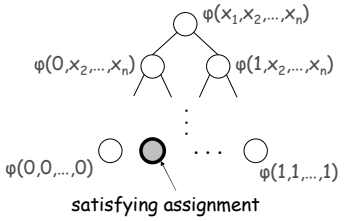
Sparse languages and **NP**

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

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

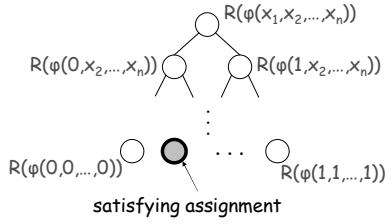
Sparse languages and **NP**

- self-reduction tree for φ:



Sparse languages and **NP**

- Applying reduction R:



Sparse languages and **NP**

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

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 :**N** → **N**

Nondeterministic space

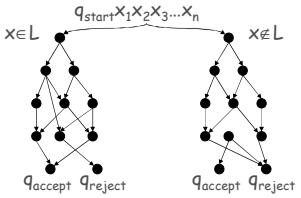
- Robust nondeterministic space classes:

NL = **NSPACE(log n)**

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

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:



Reachability

- Conclude: **NL** ⊂ **P**
  - and **NPSPACE** ⊂ **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.

Reachability

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

Two startling theorems

- Strongly believe **P** ≠ **NP**
- nondeterminism seems to add enormous power
- for space: Savitch '70:

NPSPACE = PSPACE

and

NL ⊂ **SPACE**($\log^2 n$)

Two startling theorems

- Strongly believe **NP ≠ coNP**
- seems impossible to convert existential into universal
- for space: Immerman/Szelepscényi '87/'88:

NL = coNL

Savitch's Theorem

Theorem: STCONN ∈ **SPACE(log² n)**

- Corollary: **NL ⊂ SPACE(log²n)**
- Corollary: **NPSPACE = PSPACE**

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) ∈ E )          /* base case */
 for z in V
    if PATH(x, z, i-1) and PATH(z, y, i-1) return(true);
 return(false);
end
```

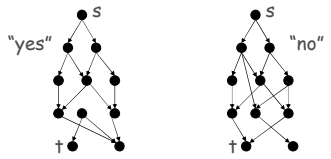

Proof of Theorem

– 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

I-S Theorem

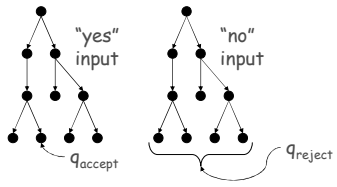
Theorem: ST-NON-CONN ∈ **NL**

- 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 ∈ 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

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

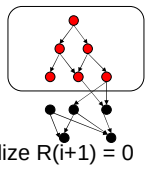
– computation paths with "bad guesses" all lead to reject

I-S Theorem

– Outline: in n phases, compute
  R(1), R(2), R(3), … 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.

I-S Theorem

– computing R(i+1) from R(i):



R(i) = R(2) = 6

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

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 ∈ V *guess* if reachable in ≤ i steps; *guess* path to u; counter++
  • KEY: if counter ≠ R(i), reject
  • at this point: **can be sure v not reachable**

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

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

Summary

- unary languages not **NP**-complete unless **P** = **NP**
  - true for sparse languages as well (homework)
- nondeterministic space classes

  NL and **NPSPACE**
- ST-CONN **NL**-complete

Summary

- Savitch: **NPSPACE** = **PSPACE**
  - Proof: ST-CONN ∈ **SPACE($\log^2 n$)**
  - open question:

    NL = **L**?
- Immerman/Szelepcsényi : **NL** = **coNL**
  - Proof: ST-NON-CONN ∈ **NL**