DD2445 LECTURE 6

Last time we moved on to SPACE COMPLEXITY
= Amount of memory used on read-write work tapes
(read-only input tape doesn't count)

$$
\begin{aligned}
DTIME(s(n)) &\subseteq SPACE(s(n)) \\
&\subseteq NSPACE(s(n)) \\
&\subseteq DTIME(2^{O(s(n))})
\end{aligned}
$$

Configuration graph $G_{M,x}$

vertices = possible states of TM M on input x

edges = transitions

DTM: out-degree 1

NDTM: out-degree ≤ 2

Can we say more about how space complexity classes such as

PSPACE = polynomial-space computation

NPSPACE = non-det poly-space computation

L = logarithmic space computation

NL = non-det log-space computation

relate to other complexity classes that we know and love?

PROPOSITION 7 NP $\subseteq$ PSPACE

Proof – Reduce from CNFSAT.
- Check all truth value assignments in lexicographic order (linear space in size of CNF formula)
- Accept if satisfying assignment found.
- Otherwise reject once all assignments tested.

OPEN PROBLEM 8 $\boxed{NP \neq L \quad ?}$

EXAMPLE 9

Let PATH = $\{\langle G, s, t\rangle \mid \exists$ path $s \rightsquigarrow t$ in digraph $G\}$

PATH $\in$ NL

Proof If there is a path, there is one of length $\leq n = |V(G)|$

Keep counter $[\log n]$ – $\log n$ bits

Walk nondeterministically (guess next vertex and check on input tape that this is OK).

Accept if reached t before counter exceeded n

[vertex indices also require space $O(\log n)$.] $\boxed{\text{Ľ}}$

Is PATH in L ? Excellent question

Would imply L = NL (i.e., PATH is NL-complete; will be discussed later.

Interestingly [Reingold '05] proved that UNDIRECTED PATH is in L ☺ (Major result)

THEOREM 10 SPACE HIERARCHY THEOREM

[Stearns, Hartmanis & Lewis '65]

That is, before Cook-Levin

If f, g are space-constructible functions s.t. $f(n) = o(g(n))$ then

$$\mathrm{SPACE}(f(n)) \subsetneq \mathrm{SPACE}(g(n))$$

Proof Will skip this. Might be good exercise.

DEF 11 PSPACE-COMPLETENESS

L' is PSPACE-hard if $L \leq_p L'$ for every $L \in$ PSPACE. If in addition $L' \in$ PSPACE then L' is PSPACE-complete.

A (not so interesting) PSPACE-complete language

$$\boxed{\text{SPACEBOUNDEDTM} = \{ \langle M, x, 1^n \rangle \mid M \text{ accepts } x \text{ in space } n \}}$$

Proof Problem set 2.

Let's look at a more interesting problem

DEF 12 A quantified Boolean formula (QBF) is a formula on the form

$$\psi = Q_1 x_1 \; Q_2 x_2 \; Q_n x_n \; \varphi(x_1, \ldots, x_n)$$

where $Q_i \in \{\forall, \exists\}$

x_i ranges over $\{0,1\}$

φ is a DNF/CNF formula

(not necessary, and Arora-Barak don't require this)

PRENEX NORMAL FORM: all quantifiers to the left.

Can easily convert to prenex.
Can easily convert to CNF / 3-CNF (skip details)

Note QBFs have determined truth value – either true or false.

Example 13

$\forall x \exists y \; (x \wedge y) \vee (\bar{x} \wedge \bar{y})$
"for all ~~x~~ exists y s.t. $x = y$" — true

$\forall x \forall y \; (x \wedge y) \vee (\bar{x} \wedge \bar{y})$
"for all x and all y they are always equal" – false

SAT – QBF with all quantifiers $\exists$
UNSAT – QBF – " – $\forall$ (and negated CNF inside)

THM 14 [Stockmeyer & Meyer '73]
The language

$$TQBF = \{ \psi \mid \psi \text{ is a true QBF} \}$$

is PSPACE-complete

Proof ~~FBR~~ $TQBF \in PSPACE$ (sketch)

Let $\psi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \; \varphi(x_1, \ldots, x_n)$ $|\varphi| = m$

Base case: If all variable set to values, just evaluate φ in $O(m)$ time and space

Inductive step

$\forall x_i\ \psi'$

set $x_i = 0$, evaluate, save — REUSING SPACE
set $x_i = 1$, evaluate, save
$\forall x_i\ \psi'$ true iff both values true
$O(1)$ extra space

$\exists x_i\ \psi'$

similar, just check if one of $x_i = 0$ and $x_i = 1$ yields true value.

Total space usage something like $O(m+n)$

$L \in PSPACE \Rightarrow L \leq_p TQBF$

M decides L in space $s(n)$
Want to construct QBF ψ of size $O(s(n)^2)$
s.t. ψ true $\Longleftrightarrow$ M accepts x

Let $m = K \cdot s(n) =$ # bits needed to encode config of M on input x.

By Claim 5.3, $\exists$ CNF $\varphi_{M,x}$ s.t. for $C, C' \in \{0,1\}^m$ $\varphi_{M,x}(C, C') = 1$ if C and C' adjacent TM configs.

Use $\varphi_{M,x}$ to define ψ s.t. $\psi(C,C') = 1$ iff $\exists$ path $C \rightsquigarrow C'$ in $G_{M,x}$.
Plug in C_{start} and C_{accept} $\Longrightarrow$ Done!
Now for the details...

Inductive definition

$\psi_i(C, C') = 1$ iff $\exists$ path $C \rightsquigarrow C'$ of $\underline{\text{length} \leq 2^i}$

S:X

$\psi_0 = \varphi_{M,x}$

After $O(m)$ steps, get $\psi = \psi_{O(m)}$.

ATTEMPT 1

If $\exists$ path of length 2^i, then $\exists$ midpoint C'' s.t.

$$\psi_{i-1}(C, C'') \wedge \psi(C'', C')$$

Why not

$$\psi_i(C,C') := \exists C'' \; \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C') \; ?$$

__Not good__: size doubles at each step $\Rightarrow$ exponential blow-up.

Need poly-size formula!

ATTEMPT 2

these are collections of m variables each

$$\psi_i(C, C') = \exists C'' \forall D^1 \forall D^2 \Big(\big(D^1 = C \wedge D^2 = C''\big) \vee \big(D^1 = C'' \wedge D^2 = C'\big) \Rightarrow \psi_{i-1}(D^1, D^2) \Big)$$

[= and $\Rightarrow$ are just convenient shorthands.
Can convert to CNF and prenex without problems]

"There is a midpoint C'' s.t. whenever D^1 is the starting point C and D^2 is the midpoint or D^1 is the midpoint and D^2 is the endpoint C', then there is a path from D^1 to D^2 in length $\leq 2^{i-1}$. The rest is just details...

[turn]

A funny observation

Proof of Thm 14 established that anything in PSPACE reduces to TQBF via analysis of $G_{M,x}$.

But we never used out-degree 1 restriction...

So... $G_{M,x}$ could have been graph for NDTM M.

So... TQBF is NPSPACE-hard.

COROLLARY 15

$$\boxed{\text{PSPACE} = \text{NPSPACE}.}$$

Can actually prove sth slightly more precise

THEOREM 16 (SAVITCH'S THEOREM '70)

For any space-constructible $s(n) \geq \log n$

$\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)$.

Proof sketch Implement reduction in Thm 14 as recursive top-down procedure

Start with upper bound $2^{O(s(n))}$

Check for all vertices in $G_{M,x}$ if can be midpoint

$O(s(n))$ space. Recurse

$O(s(n))$ space per recursive call + $O(s(n))$ recursive calls + space reusage $\Rightarrow$ space $O(s(n)^2)$ $\boxed{\text{M}}$

PSPACE: Optimal strategies for playing games

View QBF as game

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdots \varphi(x_1, x_2, x_3, x_4, \ldots)$$

$\exists$-player wants to choose x_1 such that for any choice by $\forall$-player of x_2 the formula φ can be forced to true

$\forall$-player wants to choose x_2 such that no choice for x_3 by $\exists$-player can make φ true

$\exists$-player has winning strategy $\Longleftrightarrow$ QBF true
$\forall$-player — " — $\Longleftrightarrow$ QBF false

Can model other <u>2-player games with perfect information</u> in this way

Many such games are PSPACE-complete

Hard to see how winning strategy for 1st player could have concise description for all responses to 2nd player moves

I.e., we are arguing that it seems likely that

$$NP \neq PSPACE$$

(but this is open)

Moving on next to SUBLINEAR SPACE...

When studying logarithmic space and reducing between problems, polynomial-time reductions are no good

So powerful that the reduction can solve the problem

Clearly, we don't want reduction to be more powerful than actual algorithm
Hence, let us insist on reductions in logarithmic space.

Ok, good, but...
How can a log-space reduction compute polynomial-size output?

Two solutions:

1. Write-only output tape on which space doesn't count.
   Write once   write and move right
   Never read; never move left
2. Compute reduction bit by bit

Get equivalent definitions (good exercise to show)
We go for option 2

DEF 1

$f : \{0,1\}^* \to \{0,1\}^*$ <u>implicitly logspace computable</u> if

a) f polynomially bounded ($\exists c \forall x\ |f(x)| \leq c \cdot |x|^c$)

b) $L_f = \{ \langle x, i \rangle \mid f(x)_i = 1 \}$
$L'_f = \{ \langle x, i \rangle \mid i \leq |f(x)| \}$ are both in L

Language B is <u>logspace reducible</u> to language C, denoted $B \leq_\ell C$ if $\exists$ implicitly logspace computable f s.t. $x \in B \iff f(x) \in C$

C is <u>NL-complete</u> if $C \in NL$ and $\forall B \in NL\ \ B \leq_\ell C$.

PROPOSITION 2

1. $B \leq_\ell C$ and $C \leq_\ell D \Rightarrow B \leq_\ell D$
2. $B \leq_\ell C$ and $C \in L \Rightarrow B \in L$

Proof Not hard but needs a bit of care. See textbook.

THEOREM 3 PATH is NL-complete

Recall $\boxed{PATH = \{ \langle G, s, t \rangle \mid \exists \text{ path } s \rightsquigarrow t \text{ in digraph } G \}}$

Proof Argued PATH $\in NL$ last time.

Let B in NL decided by M in log space

Define $f(x)$ to be configuration graph $G_{M,x}$ together with $C_{start} = s$ and $t = C_{accept}$.

Representation <u>adjacency matrix</u>

1 in position (C, C') if C, C' legal transition.

size $G_{M,x}$ has $\lesssim 2^{space}$ vertices. $2^{O(\log)} = poly$ — OK.

computation given C, C', look up current state and tape contents and check that C' is one of two possible configs to follow from C.

Certificate-style definition of NL?

"For every $x \in B$ $\exists$ witness y
s.t. $M(x,y) = 1$ and M runs in logspace"

Need to be careful!

Suppose x CNF formula, y satisfying assignment

Let M look up clauses in x one by one
then look up assignments in y
check that every clause satisfied.

Proves that CNFSAT $\in$ NL !

Hence NP = NL (and P=NP) – great!

Fix: Make certificate read-once

DEF 4 Certificate-style definition of NL

C is in NL if exists deterministic TM M (verifier)
with
- read-only input tape
- read-once certificate tape [read or move right each step]
- read-write tapes with $O(\log|x|)$ space bound

s.t. $x \in C \iff \exists u \in \{0,1\}^{p(|x|)}$ s.t. $M(x,u) = 1$

(for some fixed poly p depending on C).

LEMMA 5 Definitions 4 gives exactly the same class NL

Proof Exercise (not hard but useful).

Complements of space-bounded complexity classes

$\overline{PATH} = \{\langle G, s, t\rangle \mid \text{No path } s \rightsquigarrow t \text{ in digraph } G\}$

$\overline{PATH}$ in coNL (since PATH $\in$ NL)

In fact, $\overline{PATH}$ coNL-complete (since PATH NL-complete).

Log space NDTM deciding PATH:

Just walk nondeterministically for $|V(G)|$ steps from s, reject if didn't reach t

Most computation ~~paths~~ branches might reject, but if $\exists$ path then one branch will find it.

Log space NDTM deciding $\overline{PATH}$

Walk nondet & accept if didn't reach t?

A non-starter...

How can you make sure <u>all</u> branches find path $s \rightsquigarrow t$ for a no instance of $\overline{PATH}$?!

Obviously can't be done, right? Seems clear that NL $\neq$ coNL, right? Wrong.

<u>THM 6</u> $\boxed{NL = coNL}$ Immerman '88, Szelepcsényi '87

<u>Proof</u> Show that $\overline{PATH} \in$ NL.

Same ideas yield stronger statement (which we will not prove)

<u>COR 7</u> For every space constructible $s(n) \geq \log n$ it holds that NSPACE$(s(n))$ = coNSPACE$(s(n))$

Moral: Don't trust your intuition too much regarding "obvious" truths in computational complexity theory (P vs NP, anyone?)

Proof of Thm 6

Provide read-once certificate for NL-complete language $\overline{PATH}$

Important: Read-once access to certificate

But can scan graph G as many times as wanted (but not store on work tape)

$\boxed{R(i)} := \{ v \in V(G) \mid \exists \text{ path } s \rightsquigarrow v \text{ of length} \leq i \}$

$n = |V(G)|$ denote $V(G) = \{1, 2, \ldots, n\}$

denote $r_i = |R(i)|$

Want to certify $t \notin R(n)$

Starting point: Certifying $v \in R(i)$ easy

Give vertices in path $u_0 = s, u_1, u_2, \ldots, u_{i'} = v$ for $i' \leq i$

Verification
- read vertices one by one
- keep u_j and u_{j+1} in memory - log space
- keep j in memory - log space
- at each step, check $(u_j, u_{j+1}) \in E(G)$ by scanning input tape.
- check that j never exceeds i.

Let such a certificate be denoted

$\boxed{\text{IsMember}(v, i)}$ — $v \in R(i)$

Use this to construct two other types of certificates

(A) $\boxed{\text{MEMBERSHIPEXPANSION } (i, r, r')}$

Assuming $|R(i-1)| = r'$,
proof that $|R(i)| = r$

(B) $\boxed{\text{NOTMEMBER } (v, i, r)}$

Assuming that $|R(i)| = r$
proof that $v \notin R(i)$.

Suppose we can build such read-once verifiable subcertificates. Then we're done!

We all know $R(0) = \{s\}$ and $|R(0)| = 1$

~~Suppose~~ Let $r_i = |R(i)|$.

Here is the certificate

$$\boxed{\begin{array}{l} \text{MEMBERSHIPEXPANSION } (1, r_1, 1); \\ \text{MEMBERSHIPEXPANSION } (2, r_2, r_1); \\ \text{MEMBERSHIPEXPANSION } (3, r_3, r_2); \\ \vdots \\ \text{MEMBERSHIPEXPANSION } (n, r_n, r_{n-1}), \\ \text{NOTMEMBER } (t, n, r_n) \end{array}}$$

- check each line in read-once fashion.
- keep counter i and neighbourhood size r_i
  → $\log n$ space
- finally verify nonmembership certificate
- each expansion certificate for step j is verified using stored r_{j-1} → $\log n$ space

LOG VII

(B) NotMember (v, i, r)

Suppose $R(i) = \{u_1, u_2, \ldots, u_r\}$ $u_1 < u_2 < \ldots < u_r$

Let certificate be sorted list of u_j's with membership certificates

$$\boxed{\begin{array}{ll} u_1 : & \text{IsMember}(u_1, i) \\ u_2 : & \text{IsMember}(u_2, i) \\ \vdots & \\ u_r : & \text{IsMember}(u_r, i) \end{array}} =: \boxed{\text{ListMembers}(i, r)}$$

Denote this by

<u>Verification</u>
- r is known
- go over list and read u_j
- for each u_j, check certificate of membership
- check $u_j > u_{j-1}$
- check $u_j \neq v$
- check # u_j-vertices $= r$

(A) MembershipExpansion (i, r, r')

Use auxiliary certificate $\boxed{\text{NotMemberOrNeighbour}(v, i, r')}$

Assuming $|R(i+1)| = r'$, proof that $v \notin R(i+1)$

Reuse ListMembers (i, r')

<u>Verification</u> Go over list and verify u_j's as above

For each u_j, check $u_j \neq v$

<u>and that</u> $(u_j, v) \notin E(G)$

Now we can write down

MEMBERSHIPEXPANSION (i, r, r')

as an ordered list of subcertificates for vertices $1, 2, \ldots, n$

If vertex $j \in R(i)$, the line for j is

$\boxed{j: \ \text{ISMEMBER}(j, i)}$

If vertex $j \notin R(i)$, the line for j is

$\boxed{j: \ \text{NOTMEMBERORNEIGHBOUR}(j, i-1, r')}$

which is $= \text{LISTMEMBERS}(i-1, r')$

<u>Verification</u>

- For each j, check correctness of membership or non-membership certificate
- Count total # members; check that sum is $= r$

This concludes the proof ▨

SUMMARY OF THE COURSE SO FAR:

$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$

Some inclusions <u>must</u> be strict since

- $L \subsetneq PSPACE$ (space hierarchy theorem)
- $P \subsetneq EXP$ (time hierarchy theorem)

But we don't know which... (Probably most, or even all)