Why Logarithmic Space?

PSPACE is already a large class
in general, may be EXPTIME

Less than linear space would imply not reading all input
e.g., binary search: but implies indexing, will not discuss now

What about space additional to input?
(Even stronger: Online or streaming algorithms don’t know all input at once, can’t revisit)

Can we implement with little extra memory?
need a change in our computation model

The classes L and NL

Consider Turing machine with
input tape: read-only
can detect both left and right ends
work tape: size determines space complexity

For sublinear space, will only consider this model
(for space ≥ n, they are equivalent)

Def. L = SPACE(log n): decidable in log space on deterministic TM

NL = SPACE(log n): decidable in log space on nondeterministic TM

log n bits: enough for index/pointer into input string
⇒ class L = maintaining a constant number of pointers

Example: 0^k1^k

Our solution so far: zig-zag and cross out, time O(n^2) or O(n log n)

We also discussed converting number to binary
this only needs O(log n) extra space!

Compute number of zeroes on work tape (in binary).
Subtract when counting ones.

Time complexity? still O(n log n)

PATH: the prototypical NL problem

PATH = {⟨G,s,t⟩ | G is directed graph that has an s ⇝ t path }

We’ve seen PATH is in P, using linear space (BFS, DFS, etc.)

Nondeterministic solution in log space:
▶ store current node u on work tape O(log n) bits
▶ nondeterministically guess next node from successors, replace u
▶ until reaching t or running for |V| steps.

Configurations on an input word

To derive bounds, must redefine configuration structure.

Def. Let M be a TM with separate read-only input tape.
A configuration of M on w is formed of state, work tape, and the position of the two tape heads.

Input is not part of configuration (it does not change).

In M runs in f(n) space, number of work tape strings is |Γ|^f(n).
We have n positions for the input head, f(n) positions for work tape head.

Number of configurations: |Q|n^f(n)|Γ|^f(n) which is n2^{O(f(n))}
If f(n) ≥ log n, then n ≤ 2^{f(n)}, so n2^{O(f(n))} is 2^{O(f(n))}

Same argument extends condition in proof of Savitch’s theorem:
Configuration needs space log n2^{O(f(n))} = log n + O(f(n)).
If f(n) ≥ log n, this is O(f(n))
Log space reductions

We want to define the same pattern of reductions. But a polynomial-time reduction could use polynomial space! ⇒ to reduce in log space.

Def. A log space transducer $M$ is a TM with
▶ a read-only input tape
▶ a read-write work tape of $O(\log n)$ symbols
▶ a write-only output tape (output stream)

A log space computable function is a function $f : \Sigma^* \rightarrow \Sigma^*$ where $f(w)$ is the output tape contents after $M$ halts on input $w$.

$A \leq_L B$: Language $A$ is log-space reducible to language $B$ if $A$ is mapping reducible to $B$ by a log space computable function.

Many reductions can be done in log space. (e.g., PSPACE to TQBF)

NL-completeness

Def. A language $B$ is NL-complete if
1. $B \in$ NL, and
2. every $A \in$ NL is log space reducible to $B$.

Theorem: If $A \leq_L B$ and $B \in$ L, then $A \in$ L.

Can’t just map $w$ to $f(w)$, since $|f(w)|$ may be $\geq \log |w|$ ⇒ must produce $f(w)$ on demand:
$M_A$ will simulate $M_B$.
Every time machine $M_B$ needs $k^{th}$ symbol of $f(w)$, $M_A$ restarts computation of $f$, up to that symbol.

Time-space tradeoff!

Corollary If any NL-complete language is in L, then L = NL.

PATH is NL-complete

We already know PATH is in NL. Take arbitrary language $A \in$ NL.
Idea: construct graph $G$ expressing acceptance of $w$ by NTM $N_A$.
Graph nodes are configurations on input $w$.
Edge $(c_1, c_2)$ in $G$ iff can move from $c_1$ to $c_2$.
Start and (unique) accept configurations become $s$ and $t$.
Clearly $N_A$ accepts $w$ iff there is an $s \leadsto t$ path in $G$.
Can we do the reduction in log space?
Configurations take $c \log n$ space.
List nodes: generate all $c \log n$ strings, output good encodings.
Likewise, generate all possible pairs for edges.
Test that $(c_1, c_2)$ is a transition is in log space (only examine tapes at head positions in $c_1$).

NL ⊆ P

Proof: We can reduce any language in NL to PATH, and PATH \in P.
Space $f(n) \Rightarrow$ time $n^{2O(f(n))}$
Thus, log space is polynomial time.
Any language $A \in$ NL is reducible to a language in P (PATH), thus $A$ is also in P.

L ⊆ NL ⊆ P ⊆ NP ⊆ PSPACE ⊆ EXPTIME