Space Complexity

Practically important; often bottleneck, rather than time
Still use TM as model (but we’ll see some variants)

Def. Let \( M \) be a deterministic TM that halts on all inputs. The space complexity of \( M \) is the function \( f : \mathbb{N} \to \mathbb{R}^+ \), giving the maximum number of tape cells that \( M \) scans on an input of length \( n \).

We say \( M \) runs in space \( f(n) \).

For a nondeterministic TM where all branches halt on all inputs:
\[ f(n) = \max \text{ number of cells scanned on any branch, on length } n \]

Example: \textit{SAT} is linear space

\textit{SAT} is NP-complete, but it can be solved in linear space!

\begin{itemize}
  \item iterate over all truth assignments of \( x_1, \ldots, x_m \)
  \item evaluate formula on truth assignment, accept if true
  \item reject if assignments exhausted
\end{itemize}

Space? Need to store assignment: \( 10111000\ldots1101 \)
\( O(m) \) space, thus \( O(n) \) in formula size (\( m \leq n \))

Switch to next assignment: increment binary number
(no extra space)

Example: \textit{ALL}_{NFA} in NSPACE(\( n \))

A class of problems: given an acceptor (DFA/NFA/PDA etc.) and some decidable property, find space (time) complexity of checking it.
\[ \text{ALL}_{NFA} = \{ \langle M \rangle \mid M \text{ is an NFA and } L(M) = \Sigma^* \} \]

Easier to decide \textit{complement}: find string that’s \textit{not} accepted by \( M \).
Do it nondeterministically.

\( M \) is NFA: must keep set of markers to potential current states.
\( N = * \) on input \( \langle M \rangle \):
1. choose length \( k \) of string (how long? TBD)
2. repeat \( k \) times
3. nondeterministically choose a symbol and move state markers
4. if any marker on accepting state, accept
5. reject if strings exhausted

Max string length: \( 2^q \) (\( q = \) number of states of \( M \)).
\( 2^q \) marker combinations \( \Rightarrow \) longer string repeats one \( \Rightarrow \) can shorten.
Space needed: for marker (\( q \) bits) and counter (\( q \) bits) \( \Rightarrow O(n) \)

Savitch’s Theorem

\textbf{Theorem}: For any function \( f : \mathbb{N} \to \mathbb{R}^+ \), where \( f(n) \geq n \),
\[ \text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n)) \]

Anything that can be done nondeterministically in space \( f(n) \) can be done deterministically in space \( f^2(n) \).
Will see later that \( f(n) \geq \log n \) suffices.

Straightforward approach does not work:
\( f(n) \) space may run for \( O(2^{f(n)}) \) time.
Must record nondeterministic choice a \textit{each} step. \( \Rightarrow O(2^{f(n)}) \) space
A Simpler Problem: Graph Connectivity

Is there a path from \(s\) to \(t\) in graph \(G\)?

STCON: Given a graph \(G\), a node \(s\) and a node \(t\), and a length \(l\), is there a path of length \(\leq l\) from \(s\) to \(t\) in \(G\)?

Solve? BFS. Space? \(O(|V|)\) (remember visited nodes).

Can we do better?

Yes, divide and conquer!

Nondeterministically choose an intermediate node \(u\).

Check STCON \((s, u, l/2)\) and STCON \((u, t, l−l/2)\)

Space? Recursion depth is \(O(\log n)\).

Parameters: node numbers, take \(O(\log n)\) bits.

Total: \(O(\log^2 n)\).

Proof of Savitch’s Theorem

Can an NTM \(N\) get from configuration \(c_1\) to \(c_2\) in \(t\) steps with \(f(n)\) space? \(\text{CANYIELD}(c_1, c_2, t)\).

- if \(t = 1\), check if \(c_2 = c_1\) or there is an NTM transition \(c_1 \rightarrow c_2\), accept/reject accordingly.
- else \((t > 1)\), for each config. \(c_m\) of \(N\), with \(f(n)\) space
  - accept if both \(\text{CANYIELD}(c_1, c_m, t/2)\) and \(\text{CANYIELD}(c_m, c_2, t/2)\) accept
  - if all \(c_m\), exhausted, reject

Space complexity?

\(N\) has no more than \(2^{f(n)}\) configurations for some \(d\).

Recursion depth: initial \(t\) is \(2^{f(n)}\), so \(O(\log_2 2^{f(n)}) = O(f(n))\)

Stack space: for \(c_1, c_2\) and \(t\), also \(O(f(n))\). Thus \(O(f^2(n))\).

Need to know \(f(n)\) for the initial call: check 1, 2, \ldots successively

The Class PSPACE

Def. PSPACE is the class of languages that are decidable in polynomial space on a deterministic Turing machine.

\[ \text{PSPACE} = \bigcup_k \text{SPACE}(n^k) \]

By Savitch’s theorem, we have NSPACE = PSPACE.

Space-Time Constraints

At most one cell per step:

\(P \subseteq \text{PSPACE}\) and \(NP \subseteq \text{NPSPACE} = \text{PSPACE}\)

\(f(n)\) space \(\Rightarrow \leq f(n)O(2^{f(n)})\) configurations

\(\text{PSPACE} \subseteq \text{EXPTIME}\)

\(P \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXPTIME}\)