CSE200 Lecture Notes – Space Complexity

Lecture by Russell Impagliazzo
Notes by Jiawei Gao

February 23, 2016

Motivations for space-bounded computation:

- Memory is an important resource.
- Space classes characterize the difficulty of important problems
- Cautionary tale – sometimes, things don't work out like we expected.

We adopt the following conventions:

- Input tape: read only, doesn't count.
- Output tape: (for functions) write only, doesn't count.
- Work tapes: read, write, counts.

Definition 1.

$\mathsf{SPACE}(S(n))$ (or $\mathsf{DSPACE}(S(n))$) is the class of problems solvable by deterministic TM using $O(S(n))$ memory locations.

$\mathsf{NSPACE}(S(n))$ is the class of problems solvable by nondeterministic TM using $O(S(n))$ memory locations.

co-$\mathsf{NSPACE}(S(n)) = \{L \mid \overline{L} \in \mathsf{NSPACE}(S(n))\}$.

Definition 2.

$\mathsf{PSPACE} = \bigcup_k \mathsf{SPACE}(n^k)$

$\mathsf{L} = \mathsf{SPACE}(\log n)$

$\mathsf{NL} = \mathsf{NSPACE}(\log n)$

Why aren't we interested in space complexity below logaritimic space? Because on inputs of length n, we need log space to keep track of memory addresses / tape positions.

Similar as time complexity, there is a hierarchy theorem is space complexity.

Theorem 3 (Space Hierarchy Theorem)**.** If $S_1(n)$ and $S_2(n)$ are space-computable, and $S_1(n) = o(S_2(n))$, then $\mathsf{SPACE}(S_1(n)) \subsetneq \mathsf{SPACE}(S_2(n))$

The Space Hierarchy theorem is "nicer" than the Time Hierarchy Theorem, because it just needs $S_1(n) = o(S_2(n))$, instead of a log factor in the Time Hierarchy Theorem. It's proved by a straightforward diagonalization method.

1 Space complexity vs. time complexity

Definition 4. A *configuration* of M on input x includes

1. The current state;
2. Positions of the heads of input tape and work tapes;
3. Content of work tapes.

Recording a state needs constant space. Recording the positions of the heads need space n for input tape, and $S(n)$ for each work tape.

The number of contents of working tapes: $|\Gamma|^{S(n)} = 2^{O(S(n))}$.

So there are $2^{O(S(n))}$ different configurations in total.

The working process of a space-bounded TM is like a graph of all configurations. If configuration 1 can reach configuration 2 in 1 step, there is a directed edge from configuration 1 to configuration 2. There is an initial configuration, and there are accept and reject configurations. This graph is called a *configuration graph*.

For deterministic TMs, each node has only one successor. For nondeterministic TMs, each node can have arbitrary number of successors.

If M uses space $S(n)$, and has gone for more than $2^{O(S(n))}$ steps, it must be following a loop in its configuration graph, and it will loop on. Thus if we use $O(S(n))$ space to count the steps, then we can know whether M is looping, and can make it stop. Therefore, we get a time upper bound for M.

Theorem 5. $\mathsf{SPACE}(S(n)) \subseteq \mathsf{TIME}(2^{O(S(n))})$.

Corollary 6.

$\mathsf{PSPACE} \subseteq \mathsf{EXP}$.

$\mathsf{L} \subseteq \mathsf{P}$

For a language in $\mathsf{NSPACE}(S(n))$, let N be the NTM deciding it. N accepts if there's some way of accepting x. In other words, there exists a path from the start state to an accept state.

This is an instance of (s,t)-conn (read as (s,t) connectivity) problem: Given directed graph G and nodes s, t, is t reachable from s?

By DFS/BFS, (s,t)-conn is solvable in time linear to the number of nodes plus the number of edges, which is $O(2^{O(S(n))})$.

Theorem 7.

$$\mathsf{TIME}(S(n)) \subseteq \mathsf{SPACE}(S(n)) \subseteq \mathsf{NSPACE}(S(n)) \subseteq \mathsf{TIME}(2^{O(S(n))}).$$

The first inclusion is because in each step we can use at most 1 more cell, so the space complexity of any problem will not exceed its time complexity. The second inclusion straghtforwardly follows from definition. And the three inclusion is by the argument above. We also get

Corollary 8.

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}.$$

2 Savitch's Theorem

Theorem 9 (Savitch's Theorem)**.** For $S(n) \geq \log n$,

$$\mathsf{NSPACE}(S(n)) \subseteq \mathsf{SPACE}((S(n))^2).$$

Plugging in $S(n) = \log n$ and $S(n) = \mathsf{poly}(n)$, we get

Corollary 10.

$\mathsf{NL} \subseteq \mathsf{NSPACE}(\log^2 n)$

$\mathsf{NPSPACE} = \mathsf{PSPACE}$

Proof of Savitch's Theorem. We consider the graph problem: given s, t, T, is there a path of length at most T from s to t? (Let $T \leq 2^{S(n)}$.) The proof idea is to find the middle node in path, u, the path from s to u, and the path from u to t. Thus we can reuse the space. To find u, we just go through all candidates.

```
PATH(s, t, T)
    If T = 1 check if s = t, or t is a neighbor of s.
    For every u ∈ V do
        If PATH(s, u, ⌈T/2⌉) and PATH(u, t, ⌊T/2⌋) return "Yes".
    Return "No".
```

The space complexity $M(T)$ follows the recursion

$$M(T) = M(T/2) + O(\log |V|) = M(T/2) + O(S(n)).$$

The depth of the recursion is $\log T$. so $M(T) = O(S(n)) \log T$. We wanted to initialize $T \leq |V| = 2^{O(S(n))}$, so $\log T = O(S(n))$. Then $M(T) = O(S(n)) \log T = O((S(n))^2)$. □

$$PATH(s,t,T) \iff \exists u \forall b PATH(s_b, t_b, T/2) \text{ where } s_b = \begin{cases} s, \text{ if } b = 0 \\ u, \text{ if } b = 1 \end{cases} \quad \text{and } t_b = \begin{cases} t, \text{ if } b = 1 \\ u, \text{ if } b = 0 \end{cases}$$

Expanding the formula, in the end we get a formula of form

$$\exists u_1 \forall b_1 \exists u_2 \forall b_2 \ldots \exists u_{\log T} \forall b_{\log T}\ P$$

The number of quantifiers grows polynomially to the input size.

PSPACE is exactly the class of problems describable by alternating quantifiers, followed by a predicate in P. Note that unlike classes in PH, the number of quantifiers are not fixed, but a polynomial to n.

PSPACE-complete problem include

- TQBF(true quantified boolean formula): Given a boolean formula of form $Q_1 x_1 Q_2 x_2 \ldots Q_i x_i \psi(x_1 \ldots x_i)$, where $Q_1 \ldots Q_n$ are either $\exists$ or $\forall$.
- Generalization of strategy games, including "Generalized Go", "Generalized Hex" , where the space of game is not fixed. These games can be considered as one player making decisions for variables quantified by $\exists$, and the other player playing $\forall$.