Space complexity

Peter Mayr

Computability Theory, April 7, 2021

Computation may use less space than the actual input

Example

Multi-tape DTM deciding $L = \{0^k 1^k \ : \ k \in \mathbb{N}\}$:

- **Input tape:** holds input $x \in \{0,1\}^*$
- **Working tape:** Count leading 0s in binary, say k.
  Check that the last 0 is followed by k 1s and then $\sqcup$.

Required space: $O(\log |x|)$ for the computation (not counting the input).

Space complexity without input and output

The following applies to deterministic and non-deterministic machines.

Definition

A **TM** M **with input and output** is a 3-tape TM such that

- the **input tape** holds the input and is read-only,
- the **working tape** has no restrictions,
- on the **output tape** the head moves only right (write-only).

Definition

Let M be a TM with input and output that halts on any input.
M **runs in space** (has (worst case) **space complexity**) $s(n)$
if $s(n)$ is the maximum number of cells on the working tape used by M on any computational branch on any input x with $|x| = n$.

Note

For ease of comparing time and space complexity, we update our definition of running time to TMs with input & output as well.

Common classes of space complexity

Definition

$$\mathrm{DSPACE}(s(n)) := \{L(M) \;:\; M \text{ is a DTM with input \& output of space complexity } O(s(n))\}$$

$$\mathrm{NSPACE}(s(n)) := \{L(M) \;:\; M \text{ is a non-deterministic TM with input \& output of space complexity } O(s(n))\}$$

Definition

LOGSPACE := L := DSPACE($\log n$)

NLOGSPACE := NL := NSPACE($\log n$)

PSPACE := DSPACE($n^{O(1)}$) $= \bigcup_{k\in\mathbb{N}}$ DSPACE(n^k)

EXPSPACE := DSPACE($2^{n^{O(1)}}$) $= \bigcup_{k\in\mathbb{N}}$ DSPACE(2^{n^k})

Question

What about languages that can be decided in constant space?

Basic inclusions

Theorem

1. $\mathrm{DTIME}(t(n)) \subseteq \mathrm{NTIME}(t(n))$,
   $\mathrm{DSPACE}(s(n)) \subseteq \mathrm{NSPACE}(s(n))$
2. $\mathrm{DTIME}(t(n)) \subseteq \mathrm{DSPACE}(t(n))$,
   $\mathrm{NTIME}(t(n)) \subseteq \mathrm{NSPACE}(t(n))$
3. $\mathrm{NTIME}(t(n)) \subseteq \mathrm{DSPACE}(t(n))$
4. $\mathrm{NSPACE}(s(n)) \subseteq \mathrm{DTIME}(2^{O(s(n))})$ if $s(n) \geq \log n$.

Proof.

1. follows since every DTM can be considered as non-deterministic TM.
2. follows since a TM can scan only 1 tape cell in any step.

Proof 3.

Recall: A non-deterministic TM N that runs in time $t(n)$ can be simulated by a DTM M doing a breadth first search on N's computation tree.

- The input x remains unchanged on the input tape of M.
- On M's work tape we

1. first fix the current computation branch $(a_1, \ldots, a_s)$ with $a_i \leq \max(\Delta(q, a))$ and $s \leq t(n)$,
2. then simulate N's computation for that fixed choice $(a_1, \ldots, a_s)$.

Each task only requires space $O(t(n))$ on the DTM M.

Proof 4.

Let N be a non-deterministic TM with input (no output) and one working tape that runs in space $s(n)$.

Idea: Consider the configurations of N as vertices of a digraph with an edge $i \to j$ if j is a successor configuration of i. Check whether there is a path from the starting configuration `start` for input x to some accepting configuration.

Algorithm (Enumerate configurations reachable from `start`)

1. $R := \{\texttt{start}\}$ ... vertices reachable from `start`
   $B := \{\texttt{start}\}$ ... boundary of the currently reachable set
2. For $i \in B$ do
3. $\quad B := B \setminus \{i\}$
4. $\quad$ For every successor configuration j of i do
5. $\quad\quad$ If $j \notin R$, then $R := R \cup \{j\}, B := B \cup \{j\}$.
6. Return `true` if there is an acceptable configuration in R; else `false`.

Running time:

- Assume N has q states and a tape alphabet of size d. For an input of length n, there are

$$\leq qn\, s(n) d^{s(n)} = 2^{O(s(n))}$$

  configurations in N's computation tree.
- Hence the loop in 2. is executed at most $2^{O(s(n))}$ times.
- The loop in 4. is executed a constant number of times.
- Updating R, B in 5. is polynomial in $2^{O(s(n))}$.
- Hence the total running time is polynomial in $2^{O(s(n))}$.

$\square$