CS5371
Theory of Computation

Lecture 23: Complexity VIII
(Space Complexity)

Objectives

- Introduce Space Complexity
- Savitch's Theorem
- The class PSPACE

Space Complexity

Definition [for DTM]:

Let M be a DTM that halts on all inputs.

The **space complexity** of M is a function f: $N \rightarrow N$, where $f(n)$ is the maximum number of tape cells that M scans on any input of length n

If the space complexity of M is $f(n)$, we say M runs in space $f(n)$

Space Complexity (2)

Definition [for NTM]:

Let M be an NTM that all branches halt on all inputs.

The **space complexity** of M, f(n), will be the maximum number of tape cells that M scans on any branch of its computation for any input of length n

Again, if the space complexity of M is f(n), we say M runs in space f(n)

Space Complexity Classes

Definition: Let $f: N \rightarrow R$ be a function. We define two notation for describing space complexity classes as follows:

SPACE($f(n)$) = { L | L is a language decided by a **DTM** M that runs in $f(n)$ space }

NSPACE($f(n)$) = { L | L is a language decided by an **NTM** M that runs in $f(n)$ space }

Example 1

Theorem: SAT is in SPACE(n)

Proof: The following DTM M decides SAT:

M = "On input ⟨F⟩,

1. For each truth assignment,

   (a) Evaluate F on that truth assignment

2. If F is evaluated to TRUE in some case, accept. Otherwise, reject."

The space usage is O(length of ⟨F⟩). Why??

Example 2

Let ALL_{NFA} be the language

$$\{\langle M\rangle \mid M \text{ is an NFA and } L(M) = \Sigma^* \}$$

Theorem: ALL_{NFA} is in co-NSPACE(n). I.e., the complement of ALL_{NFA} is in NSPACE(n)

Note that we still do not know if ALL_{NFA} is in NP, or in co-NP.

Example 2 (cont.)

Proof Idea: We shall construct an NTM S that decides the complement of ALL_{NFA}

Question: When will an NFA M belongs to the complement of ALL_{NFA}?

Answer: ... when it rejects some string (of length at most 2^q, where q = # of states in M)

Based on this idea, the NTM S' in the next slide decides the complement of ALL_{NFA}:

Example 2 (cont.)

S' = "On input $\langle M \rangle$,

1. Place a marker on start state of NFA
2. Guess an input string w of length 2^q where q = number of states in M
3. Simulate the running of $\langle M \rangle$ on w, by updating the set of states with marker after reading a character from w
4. If at some point no accept states of M is marked, accept. Otherwise, reject."

Example 2 (cont.)

Question 1: Why is the previous decider correctly decides the complement of ALL_{NFA} ? Note that currently, only strings of length 2^q is examined…

Question 2: Is the space complexity O(length of input)?

Example 2 (cont.)

The previous NTM S' has space problem…

We now modify it a bit to give S in the next slide, which decides the complement of ALL_{NFA} in O(length of input) space:

Example 2 (cont.)

S' = "On input $\langle M \rangle$,

1. Place a marker on start state of NFA
2. Repeat 2^q times, where q = # of states in M

   (a) Guess the next input symbol and update the set of states with marker to simulate reading of that symbol

3. If at some point no accept states of M is marked, accept. Otherwise, reject."

Guess symbols one by one, instead of guess whole string at the beginning

Savitch's Theorem

Theorem: Let $f: N \rightarrow R$ be a function, with $f(n) \geq n$. Then,

$$NSPACE(f(n)) \subseteq SPACE((f(n))^2)$$

Proof: Suppose language A can be decided by an NTM in $k\,f(n)$ space, for some constant k. We shall show that it can be decided by a DTM in $O((f(n))^2)$ space

Savitch's Theorem (2)

… A naïve approach is to simulate all branches of the NTM's computation, one by one, using DTM. To do so, we need to keep track of which branch we are testing (that is, the choices made in each branch).

- Unfortunately, a branch in the NTM may have $2^{O(f(n))}$ steps (though it uses $O(f(n))$ space), so that we may need $2^{O(f(n))}$ space… NOT GOOD…

Savitch's Theorem (3)

... Instead, we solve the yieldability problem, such that given two configurations c_1 and c_2 of the NTM N, we want to decide whether c_2 can be yielded from c_1, in some number of steps

For this purpose, let us define a recursive function, called CAN_YIELD(c_1,c_2,t), the checks if c_1 can yield c_2 in t steps as follows (next slide)

Function CAN_YIELD(c_1,c_2,t) {

1. If $t = 1$, test whether $c_1 = c_2$ or whether c_1 yields c_2 in one step using the rule of NTM N. Accept if either test succeeds; Reject otherwise.
2. For each config c_m using $k\ f(n)$ space:
   a. Run CAN_YIELD(c_1,c_m,$t/2$)
   b. Run CAN_YIELD(c_m,c_2,$t/2$)
   c. If both accept, accept
3. If haven't accept yet, reject

}

Savitch's Theorem (4)

We modify N a bit, and define some terms:

- We modify N so that when it accepts, it clears the tape and moves the tape head to leftmost cell. We denote such a configuration c_{accept}
- Let c_{start} = start configuration of N on w
- Select a constant d such that N has at most $2^{d\ f(n)}$ configurations (which is the upper bound of N's running time)

Savitch's Theorem (5)

Based on this new N, there exists a DTM M that simulates N as follows:

M = "On input w,

1. Output the result

   CAN_YIELD(c_{start}, c_{accept}, $2^{d\, f(n)}$)"

Question: What is space usage of M?

Savitch's Theorem (6)

- When CAN_YIELD invokes itself recursively, it needs to store c_1, c_2, t, and the configuration c_m it is testing (so that these values can be restored upon return from the recursive call)
- Each level of recursion uses $O(f(n))$ space
- Height of recursion: $df(n) = O(f(n))$

➔ Total space = $O\left((f(n))^2\right)$

PSPACE and NSPACE

Definition: PSPACE is the class of languages that are decidable in polynomial space by a DTM. In other words,

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

Similarly, we can define NPSPACE to be the class of languages that are decidable in polynomial space by a NTM. So, what is the relationship between PSPACE and NPSPACE?

PSPACE = NPSPACE

Theorem: PSPACE = NPSPACE

Proof: By Savitch's Theorem.

PSPACE = co-NPSPACE

Theorem: PSPACE = co-NPSPACE

We first prove PSPACE $\subseteq$ co-NPSPACE:

We see that PSPACE = co-PSPACE (why?),
and co-PSPACE $\subseteq$ co-NPSPACE (why?)

We next prove co-NPSPACE $\subseteq$ PSPACE:

We see that PSPACE = co-PSPACE,
and co-NPSPACE $\subseteq$ co-PSPACE (Savitch)

P, NP, and PSPACE

Theorem: P $\subseteq$ PSPACE

Proof: If a language is decided by some DTM M in $f(n)$ time, M cannot see more than $f(n)$ cells. Thus, $\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$, so that P $\subseteq$ PSPACE

Theorem: NP $\subseteq$ PSPACE

PSPACE and EXPTIME

Theorem: PSPACE $\subseteq$ EXPTIME

Proof: If a language is decided by some DTM M in $f(n)$ space (where $f(n) \geq n$), M can visit at most $f(n)\, 2^{O(f(n))}$ configurations (why?) Thus, M must run in $f(n)\, 2^{O(f(n))}$ time.

In other words, $\mathrm{SPACE}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})$,

so that PSPACE $\subseteq$ EXPTIME

Summary

$P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$

It is shown in Chapter 9 that $P \neq EXPTIME$, so that we know at least one of the above containment ($\subseteq$) must be proper ($\subset$)

Unfortunately, at this moment, we still don't know which one(s) is proper. What most researchers believe is all are proper.

Next Time

- PSPACE-complete
- L and NL
- NL-complete