

Complexity Theory

Lecture 10

Anuj Dawar

University of Cambridge Computer Laboratory
Easter Term 2016

<http://www.cl.cam.ac.uk/teaching/1516/Complexity/>

Space Complexity

We've already seen the definition $\text{SPACE}(f)$: the languages accepted by a machine which uses $O(f(n))$ tape cells on inputs of length n . *Counting only work space.*

$\text{NSPACE}(f)$ is the class of languages accepted by a *nondeterministic* Turing machine using at most $O(f(n))$ work space.

As we are only counting work space, it makes sense to consider bounding functions f that are less than linear.

Classes

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

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

$$\text{PSPACE} = \bigcup_{k=1}^{\infty} \text{SPACE}(n^k)$$

The class of languages decidable in polynomial space.

$$\text{NPSPACE} = \bigcup_{k=1}^{\infty} \text{NSPACE}(n^k)$$

Also, define

co-NL – the languages whose complements are in NL.

co-NPSPACE – the languages whose complements are in NPSPACE.

Inclusions

We have the following inclusions:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq NPSPACE \subseteq EXP$$

where $EXP = \bigcup_{k=1}^{\infty} TIME(2^{n^k})$

Moreover,

$$L \subseteq NL \cap \text{co-NL}$$

$$P \subseteq NP \cap \text{co-NP}$$

$$PSPACE \subseteq NPSPACE \cap \text{co-NPSPACE}$$

Constructible Functions

A complexity class such as $\text{TIME}(f)$ can be very unnatural, if f is.

We restrict our bounding functions f to be proper functions:

Definition

A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is *constructible* if:

- f is non-decreasing, i.e. $f(n + 1) \geq f(n)$ for all n ; and
- there is a deterministic machine M which, on any input of length n , replaces the input with the string $0^{f(n)}$, and M runs in time $O(n + f(n))$ and uses $O(f(n))$ work space.

Examples

All of the following functions are constructible:

- $\lceil \log n \rceil$;
- n^2 ;
- n ;
- 2^n .

If f and g are constructible functions, then so are $f + g$, $f \cdot g$, 2^f and $f(g)$ (this last, provided that $f(n) > n$).

Using Constructible Functions

$\text{NTIME}(f)$ can be defined as the class of those languages L accepted by a *nondeterministic* Turing machine M , such that for every $x \in L$, there is an accepting computation of M on x of length at most $O(f(n))$.

If f is a constructible function then any language in $\text{NTIME}(f)$ is accepted by a machine for which all computations are of length at most $O(f(n))$.

Also, given a Turing machine M and a constructible function f , we can define a machine that simulates M for $f(n)$ steps.

Establishing Inclusions

To establish the known inclusions between the main complexity classes, we prove the following, for any constructible f .

- $\text{SPACE}(f(n)) \subseteq \text{NSPACE}(f(n))$;
- $\text{TIME}(f(n)) \subseteq \text{NTIME}(f(n))$;
- $\text{NTIME}(f(n)) \subseteq \text{SPACE}(f(n))$;
- $\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{\log n} + f(n))$;

The first two are straightforward from definitions.

The third is an easy simulation.

The last requires some more work.

Reachability

Recall the **Reachability** problem: given a *directed* graph $G = (V, E)$ and two nodes $a, b \in V$, determine whether there is a path from a to b in G .

A simple search algorithm solves it:

1. mark node a , leaving other nodes unmarked, and initialise set S to $\{a\}$;
2. while S is not empty, choose node i in S : remove i from S and for all j such that there is an edge (i, j) and j is unmarked, mark j and add j to S ;
3. if b is marked, accept else reject.

We can use the $O(n^2)$ algorithm for [Reachability](#) to show that:

$$\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{\log n + f(n)})$$

for some constant k .

Let M be a nondeterministic machine working in space bounds $f(n)$.

For any input x of length n , there is a constant c (depending on the number of states and alphabet of M) such that the total number of possible configurations of M within space bounds $f(n)$ is bounded by $n \cdot c^{f(n)}$.

Here, $c^{f(n)}$ represents the number of different possible contents of the work space, and n different head positions on the input.

Configuration Graph

Define the *configuration graph* of M, x to be the graph whose nodes are the possible configurations, and there is an edge from i to j if, and only if, $i \rightarrow_M j$.

Then, M accepts x if, and only if, some accepting configuration is reachable from the starting configuration $(s, \triangleright, x, \triangleright, \varepsilon)$ in the configuration graph of M, x .

Using the $O(n^2)$ algorithm for **Reachability**, we get that $L(M)$ —the language accepted by M —can be decided by a deterministic machine operating in time

$$c'(nc^{f(n)})^2 \sim c'c^{2(\log n + f(n))} \sim k^{(\log n + f(n))}$$

In particular, this establishes that $\text{NL} \subseteq \text{P}$ and $\text{NPSPACE} \subseteq \text{EXP}$.

NL Reachability

We can construct an algorithm to show that the **Reachability** problem is in NL:

1. write the index of node a in the work space;
2. if i is the index currently written on the work space:
 - (a) if $i = b$ then accept, else
guess an index j ($\log n$ bits) and write it on the work space.
 - (b) if (i, j) is not an edge, reject, else replace i by j and return to (2).

Savitch's Theorem

Further simulation results for nondeterministic space are obtained by other algorithms for [Reachability](#).

We can show that [Reachability](#) can be solved by a *deterministic* algorithm in $O((\log n)^2)$ space.

Consider the following recursive algorithm for determining whether there is a path from a to b of length at most i (for i a power of 2):

$O((\log n)^2)$ space **Reachability** algorithm:

$\text{Path}(a, b, i)$

if $i = 1$ and $a \neq b$ and (a, b) is not an edge reject

else if (a, b) is an edge or $a = b$ accept

else, for each node x , check:

1. is there a path $a - x$ of length $i/2$; and
2. is there a path $x - b$ of length $i/2$?

if such an x is found, then accept, else reject.

The maximum depth of recursion is $\log n$, and the number of bits of information kept at each stage is $3 \log n$.