Computability and Complexity
COSC 4200

Space Complexity

Deterministic Time and Space Complexity

Definition

Let M be a Turing machine and $x \in \{0,1\}^*$.

1. The *running time* of M on input x is

$$\mathit{time}_M(x) = \begin{cases} \text{the number of steps} & \\ M \text{ executes on input } x & \text{if } M \text{ halts on input } x, \\ \text{before halting} & \\ & \end{cases}$$

Deterministic Time and Space Complexity

Definition

Let M be a Turing machine and $x \in \{0,1\}^*$.

1. The *running time* of M on input x is

$$time_M(x) = \begin{cases} \text{the number of steps} & \\ M \text{ executes on input } x & \text{if } M \text{ halts on input } x, \\ \text{before halting} & \\ \infty & \text{otherwise.} \end{cases}$$

Deterministic Time and Space Complexity

Definition

Let M be a Turing machine and $x \in \{0,1\}^*$.

1. The *running time* of M on input x is

$$\mathit{time}_M(x) = \begin{cases} \text{the number of steps} & \\ M \text{ executes on input } x & \text{if } M \text{ halts on input } x, \\ \text{before halting} & \\ \infty & \text{otherwise.} \end{cases}$$

2. The *space usage* of M on input x is

$$\mathit{space}_M(x) = \begin{cases} \text{the number of worktape} & \\ \text{cells } M \text{ uses on input } x & \text{if } M \text{ halts on input } x, \\ \text{before halting} & \\ \infty & \text{otherwise.} \end{cases}$$

Time- and Space-Bounded TMs

Definition

Let $s, t : \mathbb{N} \to \mathbb{N}$, and let M be a TM. Then M is

- *$t(n)$-time-bounded* if for all $x \in \{0,1\}^*$, $\mathit{time}_M(x) \leq t(|x|)$.

Time- and Space-Bounded TMs

Definition

Let $s, t : \mathbb{N} \to \mathbb{N}$, and let M be a TM. Then M is

- $t(n)$*-time-bounded* if for all $x \in \{0,1\}^*$, $time_M(x) \leq t(|x|)$.
- $s(n)$*-space-bounded* if for all $x \in \{0,1\}^*$, $space_M(x) \leq s(|x|)$.

Time- and Space-Bounded TMs

Definition

Let $s, t : \mathbb{N} \to \mathbb{N}$, and let M be a TM. Then M is

- $t(n)$*-time-bounded* if for all $x \in \{0,1\}^*$, $time_M(x) \leq t(|x|)$.
- $s(n)$*-space-bounded* if for all $x \in \{0,1\}^*$, $space_M(x) \leq s(|x|)$.
- $O(t)$*-time-bounded* if M is $f(n)$-time-bounded for some $f(n) = O(t(n))$.

Time- and Space-Bounded TMs

Definition

Let $s, t : \mathbb{N} \to \mathbb{N}$, and let M be a TM. Then M is

- $t(n)$-*time-bounded* if for all $x \in \{0,1\}^*$, $\mathit{time}_M(x) \leq t(|x|)$.
- $s(n)$-*space-bounded* if for all $x \in \{0,1\}^*$, $\mathit{space}_M(x) \leq s(|x|)$.
- $O(t)$-*time-bounded* if M is $f(n)$-time-bounded for some $f(n) = O(t(n))$.
- $O(s)$-*space-bounded* if M is $f(n)$-space-bounded for some $f(n) = O(s(n))$.

Time and Space Complexity Classes

Definition

Given functions $t, s : \mathbb{N} \to \mathbb{N}$, define the following complexity classes.

$$\begin{aligned} \mathrm{DTIME}(t) &= \{L(M) \mid M \text{ is an } O(t)\text{-time-bounded} TM\}, \\ \mathrm{DSPACE}(s) &= \{L(M) \mid M \text{ is an } O(s)\text{-space-bounded TM}\}. \end{aligned}$$

Time and Space Complexity Classes

Definition

Given functions $t, s : \mathbb{N} \to \mathbb{N}$, define the following complexity classes.

$$\begin{aligned} \mathrm{DTIME}(t) &= \{L(M) \mid M \text{ is an } O(t)\text{-time-bounded} \, TM\}, \\ \mathrm{DSPACE}(s) &= \{L(M) \mid M \text{ is an } O(s)\text{-space-bounded TM}\}. \end{aligned}$$

Then

$$\mathrm{P} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(n^k)$$

is the class of problems decidable in polynomial time and

$$\mathrm{PSPACE} = \bigcup_{k=0}^{\infty} \mathrm{DSPACE}(n^k)$$

is the class of problems decidable in polynomial space.

Time and Space Complexity Classes

We also define the exponential-time complexity classes

$$\mathrm{E} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(2^{kn}),$$

$$\mathrm{EXP} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(2^{n^k}),$$

Time and Space Complexity Classes

We also define the exponential-time complexity classes

$$\mathrm{E} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(2^{kn}),$$

$$\mathrm{EXP} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(2^{n^k}),$$

and the exponential-space complexity classes

$$\mathrm{ESPACE} = \bigcup_{k=0}^{\infty} \mathrm{DSPACE}(2^{kn}),$$

$$\mathrm{EXPSPACE} = \bigcup_{k=0}^{\infty} \mathrm{DSPACE}(2^{n^k}).$$

Time and Space Complexity Classes

We also define the exponential-time complexity classes

$$\mathrm{E} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(2^{kn}),$$

$$\mathrm{EXP} = \bigcup_{k=0}^{\infty} \mathrm{DTIME}(2^{n^k}),$$

and the exponential-space complexity classes

$$\mathrm{ESPACE} = \bigcup_{k=0}^{\infty} \mathrm{DSPACE}(2^{kn}),$$

$$\mathrm{EXPSPACE} = \bigcup_{k=0}^{\infty} \mathrm{DSPACE}(2^{n^k}).$$

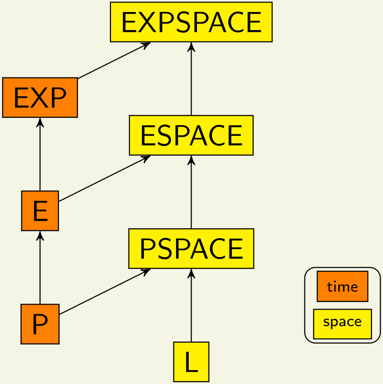
Finally, we have the logarithmic-space complexity class

$$\mathrm{L} = \mathrm{DSPACE}(\log \mathrm{n}).$$

Relationships

The following relationships among these classes are immediate.

1. $\mathrm{P} \subseteq \mathrm{E} \subseteq \mathrm{EXP}$.
2. $\mathrm{L} \subseteq \mathrm{PSPACE} \subseteq \mathrm{ESPACE} \subseteq \mathrm{EXPSPACE}$.
3. $\mathrm{P} \subseteq \mathrm{PSPACE}$; $\mathrm{E} \subseteq \mathrm{ESPACE}$; $\mathrm{EXP} \subseteq \mathrm{EXPSPACE}$.



We can prove the following inclusions of space classes in time classes.

Theorem

$\mathrm{L} \subseteq \mathrm{P}$ *and* $\mathrm{PSPACE} \subseteq \mathrm{EXP}$.

We can prove the following inclusions of space classes in time classes.

Theorem

$\mathrm{L} \subseteq \mathrm{P}$ *and* $\mathrm{PSPACE} \subseteq \mathrm{EXP}$.

The theorem follows from the following:

Lemma

If $s(n) \geq \log n$*, then*

$$\mathrm{DSPACE}(s(n)) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \cdot s(n)}).$$

In other words, running time is at most exponential in the space usage.

Proof. Let $L \in \mathrm{DSPACE}(s(n))$ and let M be a TM that decides L using at most $d \cdot s(n)$ space for some constant d. Let m be the number of states in M.

Proof. Let $L \in \mathrm{DSPACE}(s(n))$ and let M be a TM that decides L using at most $d \cdot s(n)$ space for some constant d. Let m be the number of states in M.

Then on an input x of length n, M can be in at most

$$m \cdot 3^{d \cdot s(n)} \cdot (d \cdot s(n))$$

configurations: we have m possibilities for the state, 3 possibilities (0, 1, or blank) for each of the $d \cdot s(n)$ worktape cells, and at most $d \cdot s(n)$ possibilities for the location of the tape head.

Proof. Let $L \in \mathrm{DSPACE}(s(n))$ and let M be a TM that decides L using at most $d \cdot s(n)$ space for some constant d. Let m be the number of states in M.

Then on an input x of length n, M can be in at most

$$m \cdot 3^{d \cdot s(n)} \cdot (d \cdot s(n))$$

configurations: we have m possibilities for the state, 3 possibilities (0, 1, or blank) for each of the $d \cdot s(n)$ worktape cells, and at most $d \cdot s(n)$ possibilities for the location of the tape head.

Call this number $T(n)$. Note that

$$T(n) = 2^{(\log_2 3) \cdot d \cdot s(n) + \log_2 d \cdot s(n) + \log_2 m} = O(2^{c \cdot s(n)})$$

for some constant c. We claim that M halts in at most $T(n)$ steps.

If M does not halt in $T(n)$ steps, it must repeat some configuration. But then M will come back to that configuration again and again, in an infinite loop.

If M does not halt in $T(n)$ steps, it must repeat some configuration. But then M will come back to that configuration again and again, in an infinite loop.

We know that M halts (because it decides L), so therefore it must halt within $T(n)$ steps.

If M does not halt in $T(n)$ steps, it must repeat some configuration. But then M will come back to that configuration again and again, in an infinite loop.

We know that M halts (because it decides L), so therefore it must halt within $T(n)$ steps.

Therefore M is $T(n)$-time-bounded and $L \in \mathrm{DTIME}(T(n))$. $\square$

Lemma

If $s(n) \geq \log n$, *then*

$$\mathrm{DSPACE}(s(n)) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \cdot s(n)}).$$

Theorem

$\mathrm{L} \subseteq \mathrm{P}$ *and* $\mathrm{PSPACE} \subseteq \mathrm{EXP}$.

Lemma

If $s(n) \geq \log n$, then

$$\mathrm{DSPACE}(s(n)) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \cdot s(n)}).$$

Theorem

$\mathrm{L} \subseteq \mathrm{P}$ *and* $\mathrm{PSPACE} \subseteq \mathrm{EXP}$.

Proof. Applying the lemma with $s(n) = \log n$, we have

$$\begin{aligned} \mathrm{L} &= \mathrm{DSPACE}(\log n) \\ &\subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \log n}) \end{aligned}$$

Lemma

If $s(n) \geq \log n$, *then*

$$\mathrm{DSPACE}(s(n)) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \cdot s(n)}).$$

Theorem

$\mathrm{L} \subseteq \mathrm{P}$ *and* $\mathrm{PSPACE} \subseteq \mathrm{EXP}$.

Proof. Applying the lemma with $s(n) = \log n$, we have

$$\begin{aligned} \mathrm{L} &= \mathrm{DSPACE}(\log n) \\ &\subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \log n}) \\ &= \bigcup_{c=1}^{\infty} \mathrm{DTIME}(n^c) = \mathrm{P}. \end{aligned}$$

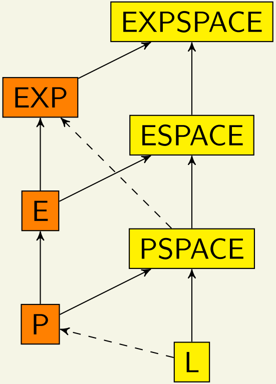
Applying the lemma with $s(n) = n^d$ for each $d \geq 1$ yields

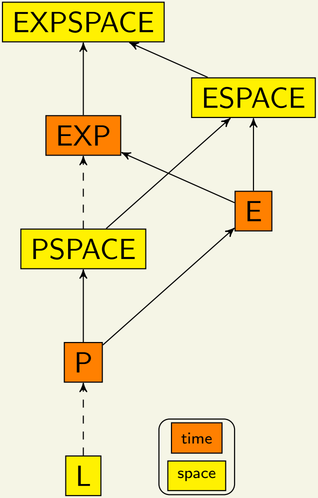
$$\begin{aligned}
\mathrm{PSPACE} &= \bigcup_{d=1}^{\infty} \mathrm{DSPACE}(n^d) \\
&\subseteq \bigcup_{d=1}^{\infty} \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{cn^d})
\end{aligned}$$

Applying the lemma with $s(n) = n^d$ for each $d \geq 1$ yields

$$\begin{aligned}
\mathrm{PSPACE} &= \bigcup_{d=1}^{\infty} \mathrm{DSPACE}(n^d) \\
&\subseteq \bigcup_{d=1}^{\infty} \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{cn^d}) \\
&\subseteq \bigcup_{d=1}^{\infty} \mathrm{DTIME}(2^{n^{d+1}}) \\
&= \mathrm{EXP} \quad \square.
\end{aligned}$$

Time and Space Classes Summary





There are several open problems about time complexity vs. space complexity:

- $\mathrm{L} = \mathrm{P}$?
  Does every problem that has a polynomial-time algorithm also have a logarithmic-space algorithm?

There are several open problems about time complexity vs. space complexity:

- $\mathrm{L} = \mathrm{P}$?
  Does every problem that has a polynomial-time algorithm also have a logarithmic-space algorithm?

- $\mathrm{P} = \mathrm{PSPACE}$?
  Does every problem that has a polynomial-space algorithm also have a polynomial-time algorithm?

There are several open problems about time complexity vs. space complexity:

- $\mathrm{L} = \mathrm{P}$?
  Does every problem that has a polynomial-time algorithm also have a logarithmic-space algorithm?

- $\mathrm{P} = \mathrm{PSPACE}$?
  Does every problem that has a polynomial-space algorithm also have a polynomial-time algorithm?

- $\mathrm{PSPACE} = \mathrm{EXP}$?
  Does every problem that has an exponential-time algorithm also have a polynomial-space algorithm?

There are several open problems about time complexity vs. space complexity:

- $\mathrm{L} = \mathrm{P}$?
  Does every problem that has a polynomial-time algorithm also have a logarithmic-space algorithm?

- $\mathrm{P} = \mathrm{PSPACE}$?
  Does every problem that has a polynomial-space algorithm also have a polynomial-time algorithm?

- $\mathrm{PSPACE} = \mathrm{EXP}$?
  Does every problem that has an exponential-time algorithm also have a polynomial-space algorithm?

- $\mathrm{E} = \mathrm{ESPACE}$? $\mathrm{EXP} = \mathrm{EXPSPACE}$?
  Does every problem that has an exponential-space algorithm also have a exponential-time algorithm?

The answer to all of these questions is probably no.

Savitch's Algorithm

The graph reachability problem is

$$\mathrm{PATH} = \left\{ \langle G, u, v \rangle \;\middle|\; G \text{ is a directed graph with a path from } u \text{ to } v \right\}$$

We know that $\mathrm{PATH} \in \mathrm{P}$ by BFS or DFS. These are $O(n)$-time algorithms that use $O(n)$ space.

Savitch's Algorithm

The graph reachability problem is

$$\mathrm{PATH} = \left\{ \langle G, u, v \rangle \;\middle|\; G \text{ is a directed graph with a path from } u \text{ to } v \right\}$$

We know that $\mathrm{PATH} \in \mathrm{P}$ by BFS or DFS. These are $O(n)$-time algorithms that use $O(n)$ space.

Theorem

$\mathrm{PATH} \in \mathrm{DSPACE}(\log^2 n)$.

Here $\log^2 n = (\log n)^2 = (\log n) \cdot (\log n)$.

Proof.
Let $G = (V, E)$, $n = |V|$, $u, v \in V$.

For any two vertices $a, b \in V$, we define the predicate $\mathit{reach}(a, b, i)$ to be true if there is a path $a \to b$ in G of length at most 2^i.

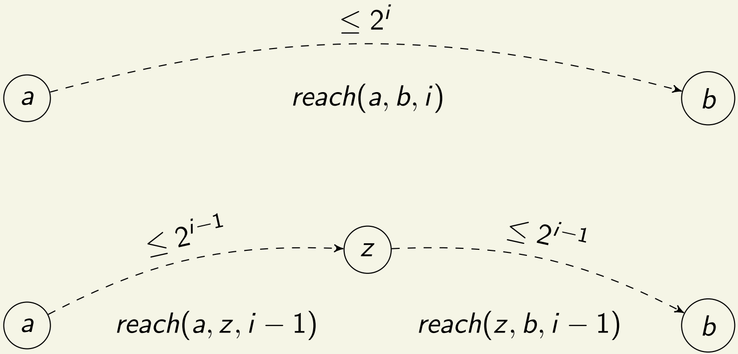
Since a shortest path in G has length at most n, we have

$$\langle G, u, v \rangle \in \mathrm{PATH} \iff \mathit{reach}(u, v, \lceil \log n \rceil)$$

reach(a, b, i) is true if there is a path $a \to b$ of length at most 2^i

Notice that

$$reach(a, b, i) \iff (\exists z)\ reach(a, z, i-1) \wedge reach(z, b, i-1).$$



We use the following algorithm to evaluate the *reach* predicate. As opposed to depth-first search or breadth-first search, this algorithm might be called middle-first search.

$REACH(a, b, i)$:
 if $a = b$ or $(a, b) \in E$, return T;
 if $i = 0$, return F;
 for all $z \in V$
  if $REACH(a, z, i-1) = T$ and $REACH(z, b, i-1) = T$
   return T;
 return F;

- On input $\langle G, u, v \rangle$, we call $REACH(u, v, \lceil \log n \rceil)$.

We use the following algorithm to evaluate the *reach* predicate. As opposed to depth-first search or breadth-first search, this algorithm might be called middle-first search.

```
REACH(a, b, i):
    if a = b or (a, b) ∈ E, return T;
    if i = 0, return F;
    for all z ∈ V
        if REACH(a, z, i − 1) = T and REACH(z, b, i − 1) = T
            return T;
    return F;
```

- On input $\langle G, u, v \rangle$, we call $REACH(u, v, \lceil \log n \rceil)$.
- The depth of the recursion tree is $\lceil \log n \rceil$.

We use the following algorithm to evaluate the *reach* predicate. As opposed to depth-first search or breadth-first search, this algorithm might be called middle-first search.

```
REACH(a, b, i):
    if a = b or (a, b) ∈ E, return T;
    if i = 0, return F;
    for all z ∈ V
        if REACH(a, z, i − 1) = T and REACH(z, b, i − 1) = T
            return T;
    return F;
```

- On input $\langle G, u, v\rangle$, we call $REACH(u, v, \lceil \log n \rceil)$.
- The depth of the recursion tree is $\lceil \log n \rceil$.
- Each *REACH* call uses $O(\log n)$ space to store a, b, i, and z.

We use the following algorithm to evaluate the *reach* predicate. As opposed to depth-first search or breadth-first search, this algorithm might be called middle-first search.

```
REACH(a, b, i):
    if a = b or (a, b) ∈ E, return T;
    if i = 0, return F;
    for all z ∈ V
        if REACH(a, z, i − 1) = T and REACH(z, b, i − 1) = T
            return T;
    return F;
```

- On input $\langle G, u, v \rangle$, we call $REACH(u, v, \lceil \log n \rceil)$.
- The depth of the recursion tree is $\lceil \log n \rceil$.
- Each *REACH* call uses $O(\log n)$ space to store a, b, i, and z.
- The total space required is $\lceil \log n \rceil \cdot O(\log n) = O(\log^2 n)$. $\square$

The algorithm in the above proof is space efficient, but very time inefficient – its running time is quasipolynomial: $O(n^{\log n})$.

This is quite a contrast to the linear-time, linear-space algorithms provided by depth-first or breadth-first search.

	time	space
BFS/DFS	$O(n)$	$O(n)$
Savitch	$O(n^{\log n})$	$O(\log^2 n)$

The algorithm in the above proof is space efficient, but very time inefficient – its running time is quasipolynomial: $O(n^{\log n})$.

This is quite a contrast to the linear-time, linear-space algorithms provided by depth-first or breadth-first search.

	time	space
BFS/DFS	$O(n)$	$O(n)$
Savitch	$O(n^{\log n})$	$O(\log^2 n)$

Open problems:

- Is there a polynomial-time, $O(\log^2 n)$-space algorithm for PATH?
- Is there a $O(\log n)$-space algorithm for PATH?
  (Is $\mathrm{PATH} \in \mathrm{L}$?)

Savitch's algorithm for graph reachability may be used to compare deterministic and nondeterministic space complexity classes.

Definition

For any $s : \mathbb{N} \to \mathbb{N}$, we define

$$\mathrm{NSPACE}(s) \quad = \quad \{L(M) \mid M \text{ is an } O(s)\text{-space-bounded NTM}\}.$$

Savitch's algorithm for graph reachability may be used to compare deterministic and nondeterministic space complexity classes.

Definition

For any $s : \mathbb{N} \to \mathbb{N}$, we define

$$\mathrm{NSPACE}(s) \;=\; \{L(M) \mid M \text{ is an } O(s)\text{-space-bounded NTM}\}.$$

Definition

A space bound $s(n)$ is *space-constructible* if the function

$$f_s(x) = \text{binary representation of } s(|x|)$$

is computable by a $O(s(n))$-space-bounded TM.

Typical space bounds like $\log n$, n, n^2, etc. are space-constructible.

Savitch's algorithm for graph reachability may be used to compare deterministic and nondeterministic space complexity classes.

Definition

For any $s : \mathbb{N} \to \mathbb{N}$, we define

$$\mathrm{NSPACE}(s) \quad = \quad \{L(M) \mid M \text{ is an } O(s)\text{-space-bounded NTM}\}.$$

Definition

A space bound $s(n)$ is *space-constructible* if the function

$$f_s(x) = \text{binary representation of } s(|x|)$$

is computable by a $O(s(n))$-space-bounded TM.

Typical space bounds like $\log n$, n, n^2, etc. are space-constructible.

Savitch's Theorem

For any space-constructible $s(n) \geq \log(n)$,

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{DSPACE}(s(n)^2).$$

Proof. Let N be an $s(n)$-spaced-bounded nondeterministic TM. Let x be an input, where $|x| = n$.

- We view the computation of N on x as a directed graph on configurations of N. There are at most $2^{c \cdot s(n)}$ configurations for some constant c.

Proof. Let N be an $s(n)$-spaced-bounded nondeterministic TM. Let x be an input, where $|x| = n$.

- We view the computation of N on x as a directed graph on configurations of N. There are at most $2^{c \cdot s(n)}$ configurations for some constant c.
- We assume that N has a unique accepting configuration F (modify N to clear its tapes if necessary).

Proof. Let N be an $s(n)$-spaced-bounded nondeterministic TM. Let x be an input, where $|x| = n$.

- We view the computation of N on x as a directed graph on configurations of N. There are at most $2^{c \cdot s(n)}$ configurations for some constant c.
- We assume that N has a unique accepting configuration F (modify N to clear its tapes if necessary).
- N on input x starts in an initial configuration C_x.
- We want to know if N can reach F from configuration C_x.

Proof. Let N be an $s(n)$-spaced-bounded nondeterministic TM. Let x be an input, where $|x| = n$.

- We view the computation of N on x as a directed graph on configurations of N. There are at most $2^{c \cdot s(n)}$ configurations for some constant c.
- We assume that N has a unique accepting configuration F (modify N to clear its tapes if necessary).
- N on input x starts in an initial configuration C_x.
- We want to know if N can reach F from configuration C_x.

We can apply the reachability algorithm on the configuration graph. This graph is exponentially large, but we do not have to explicitly write the whole thing down – the transition function of N implicitly gives us the graph.

Proof. Let N be an $s(n)$-spaced-bounded nondeterministic TM. Let x be an input, where $|x| = n$.

- We view the computation of N on x as a directed graph on configurations of N. There are at most $2^{c \cdot s(n)}$ configurations for some constant c.
- We assume that N has a unique accepting configuration F (modify N to clear its tapes if necessary).
- N on input x starts in an initial configuration C_x.
- We want to know if N can reach F from configuration C_x.

We can apply the reachability algorithm on the configuration graph. This graph is exponentially large, but we do not have to explicitly write the whole thing down – the transition function of N implicitly gives us the graph.

On input x, we run the *REACH* algorithm to see if there is a path from C_x to F. This uses

$$O\left(\log(2^{c \cdot s(n)})^2\right) = O(c^2 s(n)^2) = O(s(n)^2)$$

space. $\square$

Savitch's Theorem

For any space-constructible $s(n) \geq \log(n)$,

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{DSPACE}(s(n)^2).$$

Define the nondeterministic logspace class

$$\mathrm{NL} = \mathrm{NSPACE}(\log n).$$

Savitch's Theorem

For any space-constructible $s(n) \geq \log(n)$,

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{DSPACE}(s(n)^2).$$

Define the nondeterministic logspace class

$$\mathrm{NL} = \mathrm{NSPACE}(\log n).$$

Corollary

$\mathrm{NL} \subseteq \mathrm{DSPACE}(\log^2 n)$.

Proof. $\mathrm{NL} = \mathrm{NSPACE}(\log n) \subseteq \mathrm{DSPACE}(\log^2 n)$. $\square$

Savitch's Theorem

For any space-constructible $s(n) \geq \log(n)$,

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{DSPACE}(s(n)^2).$$

Define the nondeterministic polynomial space class

$$\mathrm{NPSPACE} = \bigcup_{c=1}^{\infty} \mathrm{NSPACE}(n^c).$$

Savitch's Theorem

For any space-constructible $s(n) \geq \log(n)$,

$$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{DSPACE}(s(n)^2).$$

Define the nondeterministic polynomial space class

$$\mathrm{NPSPACE} = \bigcup_{c=1}^{\infty} \mathrm{NSPACE}(n^c).$$

Corollary

$\mathrm{PSPACE} = \mathrm{NPSPACE}$.

Proof.

$$\mathrm{NPSPACE} = \bigcup_{c=1}^{\infty} \mathrm{NSPACE}(n^c) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DSPACE}(n^{2c}) = \mathrm{PSPACE}.$$

Since $\mathrm{PSPACE} \subseteq \mathrm{NPSPACE}$, equality follows. $\square$

Theorem

$\mathrm{NP} \subseteq \mathrm{PSPACE}$.

Proof. This is true because $\mathrm{NP} \subseteq \mathrm{NPSPACE} = \mathrm{PSPACE}$.

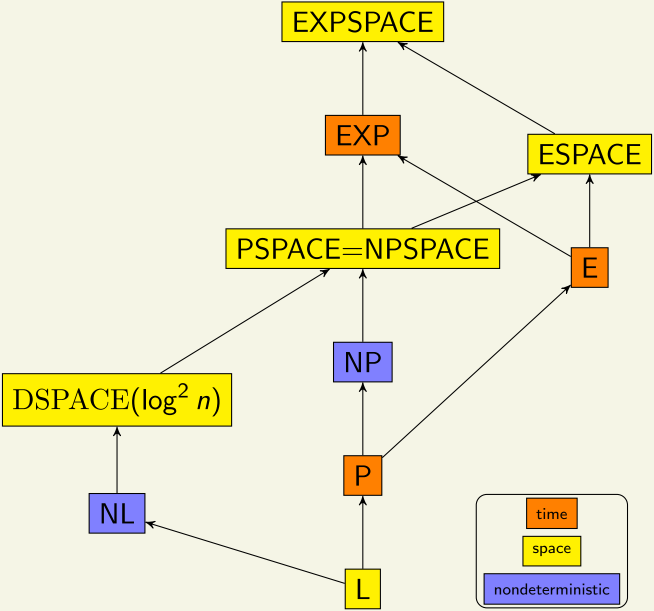
Theorem

$\mathrm{NP} \subseteq \mathrm{PSPACE}$.

Proof. This is true because $\mathrm{NP} \subseteq \mathrm{NPSPACE} = \mathrm{PSPACE}$.

For a direct proof, consider a verifier V for an NP problem A. Let M be an algorithm loops through all possible witnesses, testing each with V. If a valid witness is found, M accepts; otherwise, M rejects. Storing the current witness uses polynomial space and V uses at most polynomial space, so M uses polynomial space. $\square$

Summary


EXPSPACE
EXP
ESPACE
PSPACE=NPSPACE
E
NP
DSPACE($\log^2 n$)
P
NL
L
time
space
nondeterministic

We showed $\mathrm{L} \subseteq \mathrm{P}$. In fact, a stronger result is true:

Theorem

$\mathrm{NL} \subseteq \mathrm{P}$.

We showed $\mathrm{L} \subseteq \mathrm{P}$. In fact, a stronger result is true:

Theorem

$\mathrm{NL} \subseteq \mathrm{P}$.

Recall our lemma used to show $\mathrm{L} \subseteq \mathrm{P}$:

Lemma

If $s(n) \geq \log n$, *then* $\mathrm{DSPACE}(s(n)) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \cdot s(n)})$.

This can be extended to show:

Lemma

If $s(n) \geq \log n$, *then* $\mathrm{NSPACE}(s(n)) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \cdot s(n)})$.

We showed $\mathrm{L} \subseteq \mathrm{P}$. In fact, a stronger result is true:

Theorem

$\mathrm{NL} \subseteq \mathrm{P}$.

Recall our lemma used to show $\mathrm{L} \subseteq \mathrm{P}$:

Lemma

If $s(n) \geq \log n$, *then* $\mathrm{DSPACE}(s(n)) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \cdot s(n)})$.

This can be extended to show:

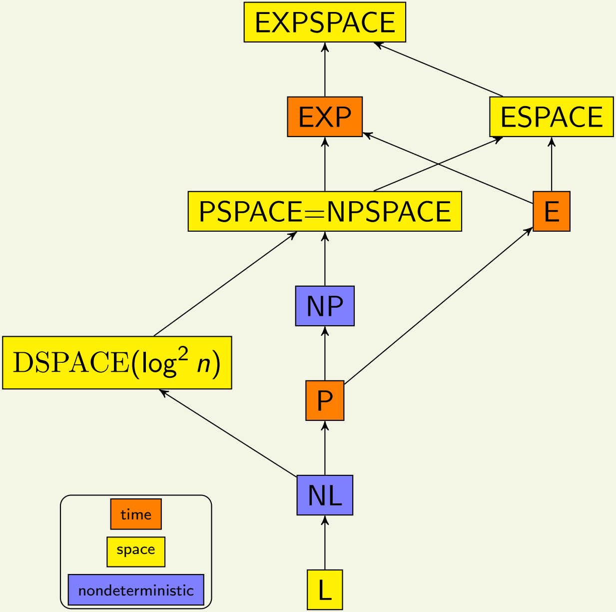
Lemma

If $s(n) \geq \log n$, *then* $\mathrm{NSPACE}(s(n)) \subseteq \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{c \cdot s(n)})$.

Proof sketch. In the nondeterministic computation tree, there are at most $2^{O(s(n))}$ configurations like before. Thus we can explore the computation tree in $2^{O(s(n))}$ time. $\square$

The theorem follows from this lemma.

Summary



Open problems:

- P = NP?
- P = PSPACE?
- NP = PSPACE?
- PSPACE = EXP?
- NP = EXP?
- NP ⊆ E? E ⊆ NP?
- PSPACE ⊆ E?
  E ⊆ PSPACE?
- L = NL?
- NL = P?
- NL = NP?
- L = NP?

Known:

- P ≠ E ≠ EXP
- L ≠ DSPACE($\log^2 n$)
  ≠ PSPACE
  ≠ ESPACE
  ≠ EXPSPACE