COMP36111: Advanced Algorithms I
Lecture 11: His Last Bow

Ian Pratt-Hartmann

Room KB2.38: email: ipratt@cs.man.ac.uk

2019–20
Outline

Motivation

Savitch’s theorem

The Immerman-Szelepcsényi Theorem

The big picture

Summary
• What we know so far:
  1. We have the following hierarchy of complexity classes

\[
\text{LogSpace} \subseteq \text{NLogSpace} \subseteq \text{PTime} \subseteq \text{NPTime} \subseteq \\
\text{PSpace} \subseteq \text{NPSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime} \subseteq \\
\text{ExpSpace} \subseteq \text{NExpSpace} \subseteq 2\text{-ExpTime} \cdots
\]

It would be nice to simplify this a bit.
  2. We have examples of complete problems for many of these:

• Horn-SAT (PTime)
• SAT (NPTime)
• QBF-SAT (PSPACE)

Moreover, we know that Krom-SAT is NLogSpace-hard and that it is in Co-NLogSpace. Can we simplify this a bit? Could it be that Krom-SAT is NLogSpace-complete?
What we know so far:

1. We have the following hierarchy of complexity classes

   \[
   \text{LogSpace} \subseteq \text{NLogSpace} \subseteq \text{PTime} \subseteq \text{NPTime} \subseteq \text{PSpace} \subseteq \text{NPSPACE} \subseteq \text{ExpTime} \subseteq \text{NExpTime} \subseteq \text{ExpSpace} \subseteq \text{NExpSpace} \subseteq 2^{\text{-ExpTime}} \ldots
   \]

   It would be nice to simplify this a bit.

2. We have examples of complete problems for many of these:

   - Horn-SAT (PTime)
   - SAT (NPTime)
   - QBF-SAT (PSPACE)

   Moreover, we know that Krom-SAT is NLogSpace-hard and that it is in Co-NLogSpace. Can we simplify this a bit? Could it be that Krom-SAT is NLogSpace-complete?
Outline

Motivation

Savitch’s theorem

The Immerman-Szelepcsényi Theorem

The big picture

Summary
• We then have the problem:

**REACHABILITY**

Given: A directed graph $G = (V, E)$ and nodes $s, t \in V$

Return: Yes if $t$ is reachable from $s$ in $G$, No otherwise.

• In an earlier lecture, we gave an algorithm showing that REACHABILITY is in $\text{TIME}(O(n))$.

• In another earlier lecture, we gave a non-deterministic procedure showing that REACHABILITY is in $\text{NLogSPACE}$. 
• The following deterministic algorithm outputs YES iff $v$ is reachable from $u$ in $G = (V, E)$ in at most $2^h$ steps:

begin isReachableNum($u$, $v$, $G$, $h$)
    if $h = 0$
        if $u = v$ or $(u, v) \in E$ return Yes
        else return No
    for $w \in V$
        if (isReachableNum($u$, $w$, $G$, $h - 1$) and isReachableNum($w$, $v$, $G$, $h - 1$)) return Yes
    return No
end

• Thus, we can then solve the reachability problem by calling isReachableNum($u$, $v$, $G$, $\lceil \log |V| \rceil$)
• To see how this works, we note that any path from $u$ to $v$ of length $\leq 2^h$ must have a midpoint $w$

![Diagram](attachment:image.png)

• Hence, there is a path from $u$ to $w$ of length $\leq 2^{h-1}$ and a path from $w$ to $v$ of length $\leq 2^{h-1}$
• How do we implement this on a Turing machine?
• Answer: by keeping the triples $\langle u, v, h \rangle$ on a work-tape:

$$\langle u, v, h \rangle$$

\[
\text{begin isReachableNum}(u,v,G,h) \\
\quad \text{if } h = 0 \\
\quad \quad \text{if } u = v \text{ or } (u, v) \in E \text{ return Yes} \\
\quad \quad \text{else return No} \\
\quad \text{for } w \in V \\
\quad \quad \text{if } (\text{isReachableNum}(u,w,G,h-1) \text{ and } \text{isReachableNum}(w,v,G,h-1)) \text{ return Yes} \\
\quad \quad \text{return No} \\
\]

• We see that this algorithm requires at most $O(h \cdot \log |V|)$ space.
• How do we implement this on a Turing machine?
• Answer: by keeping the triples $\langle u, v, h \rangle$ on a work-tape:

$$\langle u, v, h \rangle \langle u, w_1, h - 1 \rangle$$

begin isReachableNum($u, v, G, h$)
  if $h = 0$
    if $u = v$ or $(u, v) \in E$ return Yes
    else return No
  for $w \in V$
    if (isReachableNum($u, w, G, h - 1$) and
        isReachableNum($w, v, G, h - 1$)) return Yes
  return No

• We see that this algorithm requires at most $O(h \cdot \log |V|)$ space.
• How do we implement this on a Turing machine?
• Answer: by keeping the triples $\langle u, v, h \rangle$ on a work-tape:

$$\langle u, v, h \rangle \langle u, w_1, h - 1 \rangle \langle u, w_2, h - 2 \rangle$$

begin isReachableNum($u,v,G,h$)
    if $h = 0$
        if $u = v$ or $(u, v) \in E$ return Yes
        else return No
    for $w \in V$
        if (isReachableNum($u,w,G,h - 1$) and
            isReachableNum($w,v,G,h - 1$)) return Yes
    return No

• We see that this algorithm requires at most $O(h \cdot \log |V|)$ space.
• How do we implement this on a Turing machine?
• Answer: by keeping the triples $\langle u, v, h \rangle$ on a work-tape:

$$\langle u, v, h \rangle \langle u, w_1, h - 1 \rangle \langle u, w_2, h - 2 \rangle \cdots \langle u, w_\ell, h - \ell \rangle$$

begin isReachableNum($u, v, G, h$)
    if $h = 0$
        if $u = v$ or $(u, v) \in E$ return Yes
        else return No
    for $w \in V$
        if (isReachableNum($u, w, G, h - 1$) and
            isReachableNum($w, v, G, h - 1$)) return Yes
    return No

• We see that this algorithm requires at most $O(h \cdot \log |V|)$ space.
How do we implement this on a Turing machine?

Answer: by keeping the triples $⟨u, v, h⟩$ on a work-tape:

$$⟨u, v, h⟩⟨u, w_1, h − 1⟩⟨u, w_2, h − 2⟩ \cdots ⟨w_ℓ, v, h − ℓ⟩$$

begin $\text{isReachableNum}(u,v,G,h)$
  if $h = 0$
    if $u = v$ or $(u, v) \in E$ return Yes
    else return No
  for $w \in V$
    if ($\text{isReachableNum}(u,w,G,h − 1)$ and $\text{isReachableNum}(w,v,G,h − 1)$) return Yes
  return No

We see that this algorithm requires at most $O(h \cdot \log |V|)$ space.
• Hence the call \texttt{isReachableNum}(u,v,G,\lceil\log|V|\rceil) requires \(O(\log^2 |V|)\) space.

• This proves:

\textbf{Theorem (Savitch, first form)}

\textit{REACHABILITY is in SPACE}(\log^2 n).
• Suppose we have a Turing machine $M$ over an alphabet with $c$ symbols, having just one tape, and running in space $f(n)$.

• By a configuration of $M$ we mean a triple $\langle s, w, i \rangle$, where:
  • $s$ is a state of $M$;
  • $w$ is a word over the alphabet of $M$ (tape contents);
  • $1 \leq i \leq |w|$ (head position).

• Writing $w = a_1 \ldots a_\ell$, we can conveniently encode this configuration on a second (work-) tape as

$$a_1 \ldots a_{h-1}sa_h \ldots a_\ell$$

• We can think of this as the label of a node in a graph, $G$. 
• Consider again the configuration

\[ a_1 \ldots a_{h-1} s a_h \ldots a_\ell \]

and suppose \( s, a_h \rightarrow b, \text{right}, t \) is a transition of \( M \), where \( a = a_h \).

• Then we can easily compute the subsequent configuration

\[ a_1 \ldots a_{h-1} b t a_{h+1} \ldots a_\ell \]

(on another tape if you like).

• We can think of any pair of such configurations as an edge in \( G \)
• Determining whether $M$ has an accepting run (in space $f(n)$) now amounts to determining whether there is a path from the initial state of $M$ to any accepting state of $M$

\[ s^* Y \]

• The number of nodes of $G$ is bounded by $S \cdot f(n) \cdot c^{f(n)}$, where $S$ is the number of states and $c$ the size of the alphabet;

• Note that we can compute the edges of the $G$ on the fly: we often do not need the whole graph.
Theorem (Savitch (second form))

If $f$ is a proper complexity function and $f(n) \geq \log n$, then $\text{NSpace}(f) \subseteq \text{Space}(f^2)$.

Proof.
Suppose $P$ is a problem in $\text{NSpace}(f)$. Let $M$ be a nondeterministic TM running in $\text{Space}(f)$, and accepting $P$. To determine whether $x \in P$, determine whether configuration graph of $M$ has a path of length at most $2^{O(f(|x|))}$ from the initial node to an accepting node. This can be done in $\text{Space}(O(f(n))^2) = \text{Space}(O(f(n)^2))$.

Corollary

$\text{NPSPACE} = \text{PSPACE}; \text{NEXPSPACE} = \text{EXPSPACE}, \ldots$

Corollary

$\text{NPSPACE} = \text{Co-NPSPACE}; \text{NEXPSPACE} = \text{Co-NEXPSPACE}, \ldots$
Outline

Motivation

Savitch’s theorem

The Immerman-Szelepcsényi Theorem

The big picture

Summary
• The obvious non-deterministic procedure for REACHABILITY shows that this problem is in $\text{NSpace}(\log n)$.

• Applying Savitch’s theorem, we have that REACHABILITY is in $\text{Space}((\log n)^2)$.

• We now present a very simple algorithm to show that, with non-determinism, we can get the space requirements down still further.
• It was very easy to see that REACHABILITY is in NLogSpace.
• However, let us now consider its converse:

**UNREACHABILITY**
Given: A directed graph \( G = (V, E) \) and nodes \( s, t \in V \)
Return: Yes if \( t \) is **not** reachable from \( s \) in \( G \), No otherwise.

• We shall now show that UNREACHABILITY is in NLogSpace too.
• Fix a directed graph \( G = (V, E) \), and a node \( u \in V \).
• The trick is to use a very simple non-deterministic subroutine:

\[
\text{begin reachableLossy}(u,v,k) \\
\text{set } u' := u \\
\text{until } k = 0 \\
\text{guess any node } v' \\
\text{if } u' \neq v' \text{ and } (u', v') \notin E \text{ return No} \\
\text{set } u' := v' \\
\text{decrement } k \\
\text{if } u' = v \text{ return Yes} \\
\text{return No}
\]

• \( \text{reachableLossy}(u,v,k) \) has a run returning Yes iff \( v \) is reachable from \( u \) in \( k \) or fewer steps.
• Nothing is said about runs of \( \text{reachableLossy}(u,v,k) \) returning No.
Assume we have an algorithm `isReachableFail(u, v, k)` which, for $1 \leq k < n$, either returns `⊥`, Yes or No:

- `isReachableFail` has a run returning Yes, iff $v$ is reachable from $u$ in at most $k$ steps;
- `isReachableFail` has a run returning No, iff $v$ is not reachable from $u$ in at most $k$ steps;

Then the following algorithm returns the number of nodes reachable from $u$ in $k$ steps or fewer, or just returns `⊥`:

```plaintext
numReachableFail(u, k)

begin
    if $k = 0$ return 1
    set $m = 0$
    for $i = 0, \ldots, n - 1$
        let $Q = isReachableFail(u, u_i, k)$
        if $Q = ⊥$, then return ⊥
        if $Q = Yes$, then increment $m$
    return $m$
end
```
Now for the definition of $\text{isReachableFail}$ (assume $1 \leq k < n$):

begin $\text{isReachableFail}(u, v, k)$
  let $s = \text{numReachableFail}(u, k - 1)$
  if $s = \bot$ then return $\bot$
  let $m = 0$
  for $i = 0, \ldots, n - 1$
    if $\text{reachableLossy}(u, u_i, k - 1) = \text{Yes}$
      if $u_i = v$ or $(u_i, v) \in E$ then return $\text{Yes}$
      increment $m$
  if $m < s$ then return $\bot$
return $\text{No}$
Now for the our non-deterministic algorithm accepting UNREACHABILITY:

\[
\begin{align*}
\text{begin isUnreachable}(u, v, (V, E)) & \\
& \quad \text{if isReachableFail}(u, v, |V| - 1) = \text{No} \text{ then return Yes} \\
& \quad \text{return No}
\end{align*}
\]

It is easy to see that this algorithm requires only logarithmic space, and has a run returning Yes if and only if \(v\) is not reachable from \(u\) in \(G = (V, E)\).
Theorem (Immerman-Szelepcsényi, first form)

UNREACHABILITY \textit{is in} \textsf{NLogSpace}.
Theorem (Immerman-Szelepcsényi (second form))

If \( f \) is a proper complexity function and \( f(n) \geq \log(n) \), then
\[
\text{NSpace}(f) = \text{Co-NSpace}(f).
\]

**Proof.**
Suppose \( P \) is a problem in \( \text{NSpace}(f) \), and let \( \overline{P} \) be its complement problem. Let \( M \) be a nondeterministic TM running in \( \text{Space}(f) \), and accepting \( P \), and let \( x \) be an input string. Denote by \( G \) be the configuration graph of \( M \) with input \( x \). Then \( x \) is a positive instance of \( \overline{P} \) if and only if the node of \( G \) representing a successful run is unreachable from the start node of \( G \).

\( \square \)

**Corollary**

\( \text{NLogSpace} = \text{Co-NLogSpace} \).
Corollary

*KROM-SAT is $\text{NLogSpace}$-complete.*

Proof.
We showed above that KROM-SAT is in $\text{Co-NLogSpace}$ and is $\text{NLogSpace}$-complete.

But the Immerman-Szelepcsényi Theorem tells us that $\text{Co-NLogSpace}$ and $\text{NLogSpace}$ are the same thing.
Outline

Motivation

Savitch’s theorem

The Immerman-Szelepcsényi Theorem

The big picture

Summary
• We already knew that we have the following hierarchy of complexity classes:

\[
\text{LogSpace} \subseteq \text{NLogSpace} \subseteq \text{PTime} \subseteq \text{NPTime} \subseteq \text{PSpace} \subseteq \text{NPSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime} \subseteq \text{ExpSpace} \subseteq \text{NExpSpace} \subseteq 2^{-\text{ExpTime}} \cdots
\]

• Savitch’s theorem simplifies this to:

\[
\text{LogSpace} \subseteq \text{NLogSpace} \subseteq \text{PTime} \subseteq \text{NPTime} \subseteq \text{PSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime} \subseteq \text{ExpSpace} \subseteq 2^{-\text{ExpTime}} \cdots
\]

• We mentioned earlier that \text{NLogSpace} \subsetneq \text{NPSPACE}, whence by Savitch’s Theorem, \text{NLogSpace} \subsetneq \text{PSPACE}.

• Hence at least one of the inequalities

\[
\text{NLogSpace} \subseteq \text{PTime} \subseteq \text{NPTime} \subseteq \text{PSpace}.
\]

is strict; but it is not known which.
• How do the complements of these classes fit into the picture?
• We have already established the following:
  • deterministic classes (time or space) are always equal to their complement classes;
  • non-deterministic space classes from NPSpace upwards are equal to their deterministic variants (Savitch) and hence to their complement classes;
  • \textsc{NLogSpace} is equal to its complement class (Immerman-Szelepcsenyi).
• We do not know whether common non-deterministic time classes, such as \textsc{NPTIME}, \textsc{NExpTime} etc., are equal to their complements.
Outline

Motivation

Savitch’s theorem

The Immerman-Szelepcsényi Theorem

The big picture

Summary
Summary

- In this lecture, we have proved:
  - Savitch's theorem
  - the Immerman-Szelepcsény theorem.

- The former implies that $\text{NSPACE}(g) = \text{SPACE}G$ for ‘large’ $G$, allowing us to ignore most non-deterministic space classes.

- It follows that these classes are closed under complementation.

- The latter tells us that even the ‘small’ class $\text{NLogSPACE}$ is closed under complementation.

- Reading for this lecture:
  - Sipser, Ch. 8 (Space Complexity). You need not read “Winning strategies for games” or “Generalized geography”.