COMP36111: Advanced Algorithms I
Lecture 9: Savitch’s Theorem and the Immerman-Szelepcsényi Theorem

Ian Pratt-Hartmann

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

2017–18
• Reading for this lecture:
  • Sipser, Ch. 8 (Space Complexity).
Outline

REACHABILITY again

Savitch’s theorem

The Immerman-Szelepcsényi Theorem

Standard hierarchy

Time and space

The big picture
• Recall that a *directed graph* is a pair $G = (V, E)$, where $V$ is a set and $E \subseteq \binom{V}{2}$ a set of ordered pairs of distinct elements of $V$.

• We assume $G$ is encoded on the input tape of a TM as follows:

$$7 : 0, 1; 0, 2; 0, 3; 1, 0; 1, 3; 1, 4; 2, 5; 3, 6; 5, 1; 7, 2; 7, 5; 7, 6$$
If $G = (V, E)$ is a directed graph, and $u, v \in V$, we say that $v$ is *reachable* from $u$ if there exists a sequence $u = u_0, \ldots, u_m = v$ from $V$ with $m \geq 0$ such that, for each $i$ ($0 \leq i < m$) $(u_i, u_{i+1}) \in E$.

In our graph $G$, $v_6$ is reachable from $v_0$

since we have the sequence

$$v_0 \rightarrow v_1 \rightarrow v_3 \rightarrow v_6.$$
• However, $v_7$ is not reachable from $v_0$. 
• 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 $\text{TIME}(O(n))$.

• A bit of revision won’t hurt . . .
• An algorithm for solving REACHABILITY:

begin DFS-directed((V, E), u, v)
  DFS-aux((V, E), u)
  if v is marked
    return Y
  return N
end DFS-directed

begin DFS-aux((V, E), u)
  mark u
  for each e ∈ edges(u) do
    if e = (u, w) with w unmarked do
      DFS-aux((V, E), w)
  end DFS-aux

• We saw an essentially identical algorithm in Lecture 1a. It runs in linear time (and hence linear space).
Outline

REACHABILITY again

Savitch’s theorem

The Immerman-Szelepcsényi Theorem

Standard hierarchy

Time and space

The big picture
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

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 $w$

• 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:

\[
\langle u, v, h \rangle
\]

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

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 \( \langle u, v, h \rangle \) on a work-tape:

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

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 \( (\text{isReachableNum}(u,w,G,h - 1) \) 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.
• 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 time $f(n)$) now amounts to determining whether there is a path from the initial state of $M$ to any accepting state of $M$.

• 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 $\text{Space}(O(f(n))^2)$.

Corollary
$\text{NPSpace} = \text{PSpace}; \text{NExpSpace} = \text{ExpSpace}, \ldots$

Corollary
$\text{NPSpace} = \text{Co-NPSpace}; \text{NExpSpace} = \text{Co-NExpSpace}, \ldots$
Outline

REACHABILITY again

Savitch’s theorem

The Immerman-Szelepcsényi Theorem

Standard hierarchy
  Time and space
  The big picture
• The algorithm DFS-directed showed that REACHABILITY is in $\textsc{Time}(n)$ and hence in $\textsc{Space}(n)$.
• The algorithm $\text{isReachableNum}$ showed that REACHABILITY is in $\textsc{Space}((\log n)^2)$.
• We now present a very simple algorithm to show that, with non-determinism, we can get the space requirements down still further.
• The following non-deterministic algorithm has a run which outputs YES iff \( v \) is reachable from \( u \) in \( G = (V, E) \):

begin isReachable\((u,v,G)\)
  let current = \( u \)
  let counter = 0
  until current = \( v \) or counter = \( |V| \)
    guess a node \( u' \in V \)
    if \((current, u') \notin E\) return NO
    let current = \( u' \)
    increment counter
  if current = \( v \) return YES
  else return NO

• The only things we need to store are current and counter, which require only \( \log |V| \) bits.

• Thus we see that REACHABILITY is in \( \mathsf{NSpace}(\log n) \) —i.e. \( \mathsf{NLogSpace} \).
Theorem

REACHABILITY is $\text{NLogSpace}$-complete.

Proof.

We showed above that REACHABILITY is in $\text{NLogSpace}$.

Suppose $L$ is a language recognized by a non-deterministic TM, $M$, running in time $O(\log n)$. Given an input $x$, let $G$ be the configuration graph for $M$ with input $x$. Let $u$ be the node representing the initial configuration. We may assume this graph has a single accepting node $v$. Now, $x \in L$ if and only if $(G, u, v)$ is an instance of REACHABILITY. The mapping $x \mapsto (G, u, v)$ can easily be constructed in space bounded by $\log n$. \hfill \square
• It was very easy to see that \textsc{Reachability} is in \textsc{NLogSpace}.

• However, let us now consider its converse:

\begin{center}
\textbf{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.
\end{center}

• We shall now show that \textsc{Unreachability} is in \textsc{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:

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

• 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 reachableLossy$(u, v, k)$ returning No.
• Assume we have an algorithm \texttt{isReachableFail}(u, v, k) which, for $1 \leq k < n$, either returns \texttt{\bot}, Yes or No:
  • \texttt{isReachableFail} has a run returning Yes, iff $v$ is reachable from $u$ in at most $k$ steps;
  • \texttt{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 \texttt{\bot}:

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

\begin{verbatim}
begin isReachableFail(u, v, k)
    let s = numReachableFail(u, k - 1)
    if s = ⊥ then return ⊥
    let m = 0
    for i = 0, ..., n - 1
        if reachableLossy(u, u_i, k - 1) = Yes
            if u_i = v or (u_i, v) ∈ E then return Yes
                increment m
        if m < s then return ⊥
    return No
\end{verbatim}
• Now for the our non-deterministic algorithm accepting UNREACHABILITY:

\[
\text{begin isUnreachable}(u, \nu, (V, E))
\]

if isReachableFail\((u, \nu, |V| - 1) = \text{No}\) then return Yes
return No

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

UNREACHABILITY is in $\text{NLogSpace}$. 
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 \( G \).

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

Corollary
\( \text{KROM-SAT} (= 2\text{-SAT}) \) is in \( \text{NLogSpace} \).
Theorem

The problem \( \text{KROM-SAT} \) \( (\equiv 2\text{-SAT}) \) is \( \text{NLogSpace-complete} \).

Proof.

We have just shown that KROM-SAT is in \( \text{NLogSpace} \). For \( \text{NLogSpace} \)-hardness, we reduce the problem \( \text{UNREACHABILITY} \) \( (\equiv \text{Co-REACHABILITY}) \) to KROM-SAT. Let \( G = (V, E) \) be a directed graph, and \( u_0, v_0 \in V \) be vertices. Treating the vertices \( V \) as propositional variables, define the set of clauses \( \Gamma_G \)

\[
\{u_0, \neg v_0\} \cup \{u \rightarrow v \mid (u, v) \in E\}.
\]
Proof.
If \( u_0, \ldots, u_m = v_0 \) is a path through \( G \), then any truth-value assignment marking \( u_0 \) and \( \{ u \rightarrow v \mid (u, v) \in E \} \) true must make \( v_0 \) true. Hence \( \Gamma_G \) is unsatisfiable.

Conversely, if \( v_0 \) is not reachable from \( u_0 \), let \( \theta \) be the truth-value assignment

\[
\theta(v) = \begin{cases} 
  T & \text{if } v \text{ is reachable from } u_0 \text{ in } G; \\
  F & \text{otherwise.}
\end{cases}
\]

Then \( \theta \) evidently satisfies \( \Gamma_G \).

\( \text{NLogSpace} \)-hardness of KROM-SAT then follows from the \( \text{NLogSpace} \)-hardness of UNREACHABILITY.
Outline

REACHABILITY again

Savitch's theorem

The Immerman-Szelepcsényi Theorem

Standard hierarchy
  Time and space
  The big picture
• Consider again the classes \( \text{TIME}(f) \), \( \text{NTIME}(f) \), \( \text{SPACE}(f) \).
• How are these related?
• First of all, it is trivial that

\[
\text{TIME}(f) \subseteq \text{NTIME}(f).
\]

• What about \( \text{NTIME}(f) \) and \( \text{SPACE}(f) \)?
  • Consider a TM, \( M \), running in \( \text{NTIME}(f) \), and acting on input \( x \) of length \( n \).
  • \( M \) can make at most \( f(n) \) non-deterministic choices.
  • Represent these choices as a digits as a string \( s \) of length \( f(n) \).
  • We can run through all possible strings, guiding the choices of \( M \) using \( s \), succeeding if \( M \) ever succeeds.
  • The resulting deterministic TM takes no more space, and recognizes the same language as \( M \).

• Thus, we have

\[
\text{NTIME}(f) \subseteq \text{SPACE}(f).
\]
• To relate $\text{SPACE}(f)$ to a time-complexity class, consider a TM, $M$, running in $\text{SPACE}(f)$, and operating on an input $x$ of length $n$.

• We may suppose $M$ has $K$ tapes ($K - 2$ work-tapes) and uses an alphabet $\Sigma$.

• A configuration is given by $K - 2$ strings of length at most $f(n)$, together with one of $|Q|$ states, and a $K$-tuple of integers ($\leq f(n)$) indicating the head position on each tape.

• The total number of such configurations is $|Q|.|\Sigma|^{(K-2)f(n)+K\log(f(n))}$, and reachability in such a graph can be decided in time $c' \cdot |Q|.|\Sigma|^{(K-2)f(n)+K\log(f(n))}$ for some constant $c'$.

• Thus, we have

$$\text{SPACE}(f(n)) \subseteq \bigcup_{a>0} \text{TIME}(2^{af(n)}).$$
• Applying these results to the larger classes $\text{PTime}$, $\text{NPTime}$, $\text{PSpace}$ etc, a nice picture emerges:

$$\text{PTime} \subseteq \text{NPTime} \subseteq \text{PSpace} \subseteq \text{ExpTime} \ldots$$
• What about the non-deterministic stuff?
• We know from Savitch’s theorem that \( \text{PSPACE} = \text{NPSpace} \), \( \text{EXPSPACE} = \text{NEXPSPACE} \) etc., so we can henceforth ignore large non-deterministic space-classes.
• Warning: it does not follow from Savitch’s theorem that \( \text{LOGSPACE} = \text{NLOGSPACE} \), and we do not know whether this equation holds.
• Warning: it does not follow from Savitch’s theorem that \( \text{PTIME} = \text{NPTIME} \), etc, and we do not know whether these equations hold.
Thus, we have:

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

We showed earlier that \( \text{PTime} \subset \text{ExpTime} \), and hence that at least one of the inequalities

\[ \text{PTime} \subseteq \text{NPTime}, \quad \text{NPTime} \subseteq \text{PSpace}, \quad \text{PSpace} \subseteq \text{ExpTime} \]

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;
  • \( \text{NLogSpace} \) is equal to its complement class (Immerman-Szelepcseényi).
• We do not know whether common non-deterministic time classes, such as \( \text{NPTime} \), \( \text{NExpTime} \) etc., are equal to their complements.