So far: \textbf{Space Complexity}

\[ \text{SPACE}(S(n)) = \{ A : A \text{ is decided by a TM as above using space } O(S(n)) \} \]

\[ \text{NSPACE}(S(n)) = \{ A : A \text{ is decided by an NTM as above using space } O(S(n)) \} \]

\textbf{Example: Regular Languages} \subseteq \text{SPACE}(L)

\textbf{Note: In practice, space complexity is often as important as time complexity.}

\textbf{Theorems:}

\begin{align*}
(1) & \text{ TIME}(T(n)) \subseteq \text{SPACE}(T(n)) \\
(2) & \text{ NTIME}(T(n)) \subseteq \text{SPACE}(S(n)) \subseteq \text{TIME}(2^{O(S(n))}) \]
(3) & \text{ SPACE}(S(n)) \subseteq \text{TIME}(2^{O(S(n))}) \]

\textbf{Proof: (1)} \text{ TIME}(T(n)) \subseteq \text{SPACE}(T(n)) \text{ since a TM \textit{way} time T(n) only can touch T(n) cells.}

\textbf{NTIME}(T(n)) \subseteq \text{SPACE}(T(n)) \text{ can try all paths in tree using only O(T(n)) space.}
(b) Like algorithm for AGBM.

# of configurations on an input of length $n$ before a repeat is:

$$\leq N \cdot 2^{dS(n)}$$

proof of input $x$ copies of input $x$ work-tape cells
head $x$ work-tape head
$O(1)$

Since $S(n) \geq \log_2 n$, this is $2^{O(S(n))}$, run space bounded TM using forever and reject if it runs too long.

Total space $O(S(n))$ on memory + $O(S(n))$ bits for min.

$2^{O(S(n))}$ time & $O(S(n))$ space simultaneously

(c) Obvious algorithm would be to try the "tree of paths" of length $2^{O(S(n))}$.

Each path could only run this long but there would be too many leaves to check: exponential in $2^{O(S(n))}$

Observation: Configurations on these other paths could be repeats of one on the first path. Still only $2^{O(S(n))}$ total different configs.

For each space-bounded TM $M$ and input $x$, we define a directed graph $G_M(x)$ on configurations $G_M(x)$.
**Definition**

\( G_{M,x} \): each vertex is a configuration of \( M \) on input \( x \)

(i.e. initial state of \( M \) on input \( x \))

Start configuration: \( C_0 = (\text{q}_0, x, \text{w}) \)

Step 1:

\[ \text{edge } C \rightarrow D \text{ iff } C \rightarrow M \text{ D} \]

"Yields in one step"

\[ \text{out-degree } \leq b \]

for \( b \) nice constant depending on \( \delta \) function of \( M \)

without loss of generality, there is a unique accepting empty word

\( \text{Accept } = (\text{q}_a, x, \text{w}) \)

(simply have \( M \) clean up everything before accepting)

**Note:** \( M \) accepts \( x \) if there is a path from \( C_0 \) to \( \text{Accept} \) in \( G_{M,x} \)

(of length \( 20^{|x||w|} \))

- \( M \) deterministic \( \Rightarrow G_{M,x} \) has out-degree 1

**Now:** to prove (c): Build \( G_{M,x} \) and apply graph search for a path from \( C_0 \) to \( \text{Accept} \)

Time \( \approx \left(20^{|x||w|}\right)^2 \) which is \( 20^{|x||w|} \)
Note: The algorithm uses \( 2^{O(S(n))} \) space and \( 2^{O(S(n))} \) time.

Can we do better for simulating \( \text{NSPACE}(S(n)) \) machines?

Then (Switch)

\[ S(n) \geq \log_2 n = \text{NSPACE}(S(n)) \leq \text{SPACE}(S^2(n)) \]

Proof idea. We search for path from \( C_0 \) to \( C_{\text{accept}} \) in \( G_{\text{mix}} \) but don't write down the whole graph.

We know that if there is such a path, then it has at most

\[ T \leq 2^{O(S(n))} \]

steps for some \( d \).

Let's pretend we know \( T \):

\[ T \leq T^{1/2} \leq T^{1/2} \rightarrow C_T \]

Then some \( C_{\text{mid}} \) above exists.

Define function \( \text{CAN YIELD}_T(C, D) = \begin{cases} 
\text{true} & \text{if } C_T \text{ \text{can yield } } D \text{ \text{in } } T \text{ \text{steps}} \\
\text{false} & \text{otherwise}
\end{cases} \)
Observe that we have the following recursive properties:

\[
\text{CANYIELD}_0(C, D) \iff C = D
\]

\[
\text{CANYIELD}_1(C, D) \iff C \rightarrow D, \text{ i.e., } C \rightarrow \text{mid}
\]

\[
\text{or } C = D
\]

Algorithm: can check using & function of \( M \)

\[
\text{CANYIELD}_k(C, D) \iff \exists \text{mid} \quad \text{such that} \quad x \quad \text{such that} \quad \text{CANYIELD}_{k-1}(C, \text{mid}) \quad \text{and} \quad \text{CANYIELD}_{k-1}^*(C, \text{mid}, D)
\]

Algorithm: Try all possible \( \text{mid} \) and compute recursive calls

Goal: Compute \( \text{CANYIELD}_T(C_0, C_{\text{target}}) \)

Space used for recursive algorithm:

- \# of levels: \( \log_T n \) which is \( O(S(n)) \)

- Each level of call stack:
  \[
  O(S(n)) \quad \text{for } C, D, T
  \]

- Other space used at each call level:
  \[
  O(S(n)) \quad \text{for } \text{mid}
  \]

Total \( O(S^2(n)) \)

To do this we assumed that we knew \( T \)

But we don't actually need that.
We modify the above to try all possible
\[ T = 2^k \]
using \( S = 1, 2, \ldots, \frac{2^k}{k} \) memory cells.

Run above

Keep track of whether TM actually ever tried a rightward move when an last cell
of work tape

If \( \text{ANYNEW}(G,C,\text{new}) \) is true, accept.
If no path found but a rightward move

then unmark \( S, T \)
if no path found and no rightward move
then reject.

Total: \( O(S^2(n)) \) space & count

Note: time and is more than \( 2O(S^2(n)) \)!

It is \( 2O(S^2(n)) \) but we only focus
on space

Recall:

\[ \text{PSPACE} = \bigcup_n \text{SPACE}(n^k) \]

\[ \text{NPSPACE} = \bigcup_n \text{NPSPACE}(n^k) \]

Cor

\[ \text{NPSPACE} = \text{PSPACE} \]

Cor

\[ \text{NPSPACE}(n^k) \subseteq \text{SPACE}(n^{2k}) \]

Now \( P \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP} \)

\( P \neq \text{EXP} \) (giant leeway) but all other containments

conjectured to be \( \notin \) (open)
It $P \subseteq \text{PSPACE}$? (P $\subseteq$ PSPACE implied by P $\subseteq$ NP and potentially easier to prove)

**Def.** $B$ is **PSPACE-hard** iff 
$orall A \in \text{PSPACE}, A \leq_P B.$

**Def.** $B$ is **PSPACE-complete** iff
- $B \in \text{PSPACE}$
- $B$ is PSPACE-hard

Let $\varphi$ be a Boolean formula in variables $x_1, \ldots, x_n$

$\varphi \in \text{SAT} \iff \exists x_1, \ldots, \exists x_n \varphi(x_1, \ldots, x_n)$ is true

$\varphi \in \text{TAUT} \iff \forall x_1, \ldots, \forall x_n \varphi(x_1, \ldots, x_n)$ is true

*fully quantified Boolean formula*

**Def.** $\text{TQBF} = \exists \varphi: \varphi$ is a fully quantified Boolean formula (that is true)

*quantifiers may alternate*

Next time we prove:

Then $\text{TQBF}$ is PSPACE-complete