Theory of Computation
Time/Space Hierarchy

Barath Raghavan
CS 361 Fall 2009
Williams College

MISC

HW 6 due tomorrow
Quine page

Literals

OR

TIME and SPACE
How do we define running time?

All languages decidable by some TM in $O(t(n))$ steps.

**TIME($t(n)$)**

TM model: 3 tapes
- **Input tape** (read-only)
- **Work tape** (read-write)
- **Output tape** (write-only)

A TM M uses K space if, for any input x:

M(x) never goes beyond the Kth cell on the work tape.
**SPACE**($s(n)$)

All languages decidable by some TM in $O(s(n))$ space.

What about non-determinism?

**NTIME**($t(n)$)

All languages decidable by some non-deterministic TM in $O(t(n))$ steps.

**NSPACE**($s(n)$)

All languages decidable by some non-deterministic TM in $O(s(n))$ space.
What is an alternative definition for $\text{NTIME}(t(n))$?

How are $\text{TIME}$ and $\text{SPACE}$ related?

Proposition:

$\text{TIME}(t(n)) \subseteq \text{NTIME}(t(n))$
Proposition:
\[ \text{TIME}(t(n)) \subseteq \text{NTIME}(t(n)) \]

\[ L \in \text{TIME}(t(n)) \Rightarrow L \in \text{NTIME}(t(n)) \]

\[ \exists O(t(n)) \text{ time TM } M \text{ that decides } L \]

Run M in V.
Proposition:

\[ \text{TIME}(t(n)) \subseteq \text{SPACE}(t(n)) \]

\[ L \in \text{TIME}(t(n)) \Rightarrow L \in \text{SPACE}(t(n)) \]
\[ \exists O(t(n)) \text{ time TM } M \text{ that decides } L \]
\[ \exists O(t(n)) \text{ space TM } S \text{ that decides } L \]

Run M in S. S has no time bound and M can’t visit more than t(n) cells in t(n) time.

Proposition:

\[ \text{NTIME}(t(n)) \subseteq \text{SPACE}(t(n)) \]

\[ L \in \text{NTIME}(t(n)) \Rightarrow L \in \text{SPACE}(t(n)) \]
\[ \exists O(t(n)) \text{ time TM } V \text{ that verifies } L \]
Proposition:

\( \text{NTIME}(t(n)) \subseteq \text{SPACE}(t(n)) \)

\( L \in \text{NTIME}(t(n)) \Rightarrow L \in \text{SPACE}(t(n)) \)

\( \exists O(t(n)) \text{ time TM } V \) that verifies \( L \)

\( \exists O(t(n)) \text{ space TM } S \) that decides \( L \)

Test all certificates \( y \) of length \( t(n) \) on \( V \) in \( S \).

\( \text{SPACE} \) can be reused.
\( \text{TIME} \) can’t.

How powerful is \( \text{SPACE} \)?
Proposition:

\[
\text{SPACE}(s(n)) \subseteq \bigcup_{c>0} \text{TIME}(n \cdot 2^{cs(n)})
\]

Definition: a configuration of \( M \) is a tuple (state, input head, work head, work tape)

Definition: a configuration of \( M \) is a tuple (state, input head, work head, work tape)

\[
|Q| \cdot n \cdot s(n) \cdot |\Gamma|^s(n)
\]

configurations
Time Hierarchy Theorem

\[ g(n) = o\left(\frac{f(n)}{\log(f(n))}\right) \Rightarrow \]
\[ \text{TIME}(g(n)) \subset \text{TIME}(f(n)) \]

\[ \text{EXPTIME} = \bigcup_{c>0} \text{TIME}(2^{n^c}) \]

\[ \text{PSPACE} = \bigcup_{c>0} \text{SPACE}(n^c) \]

\[ \text{L} = \text{SPACE}(\log(n)) \]

Open questions:
- \( P \neq \text{NP} \)
- \( \text{NP} \neq \text{coNP} \)
- \( \text{PSPACE} \neq \text{EXPTIME} \)
- \( P \neq \text{PSPACE} \)
But what if?

Algorithmica
P=NP, SAT is fast.
Consequences are huge.

Impagliazzo’s worlds

Theorem proving
VLSI layout
Machine learning
If $P=NP$, creating great music takes no more work than recognizing great music. (The computer could use you as a verifier.)

It would be impossible to differentiate humans and machines. Public-key crypto gone.

(How do you do public-key crypto without big numbers?)

Heuristica $P!=NP$ but there are heuristics to solve almost everything in $NP$ fast.
Pessiland
Cryptography is broken but NP still hard.

Cryptomania
The world we’re in (maybe).

(Weirdland)
P = NP but the algorithm for SAT is very slow.

Reading: Sipser 7.5