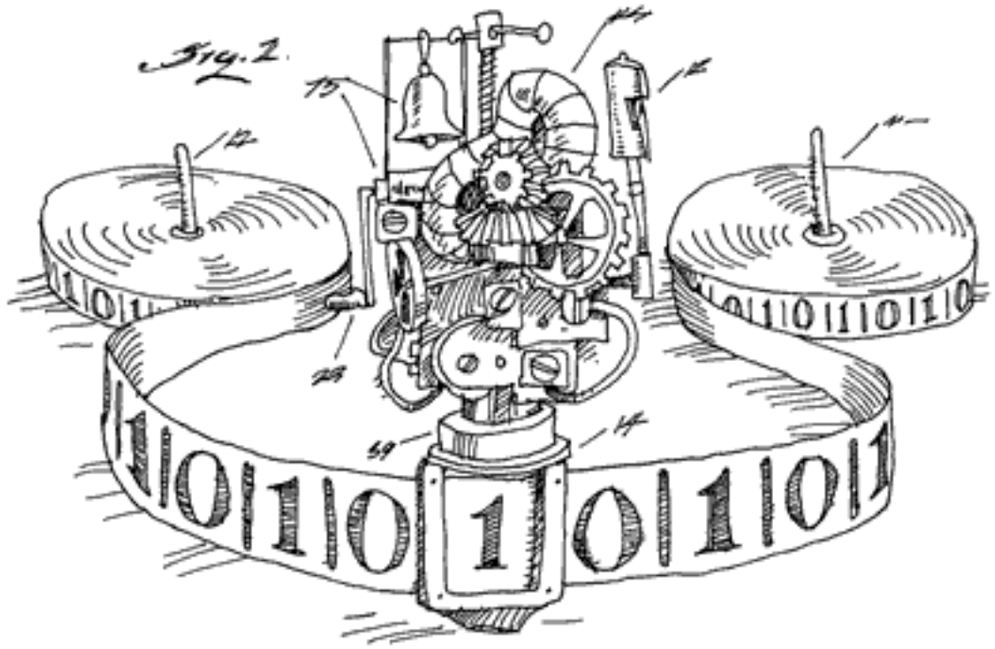
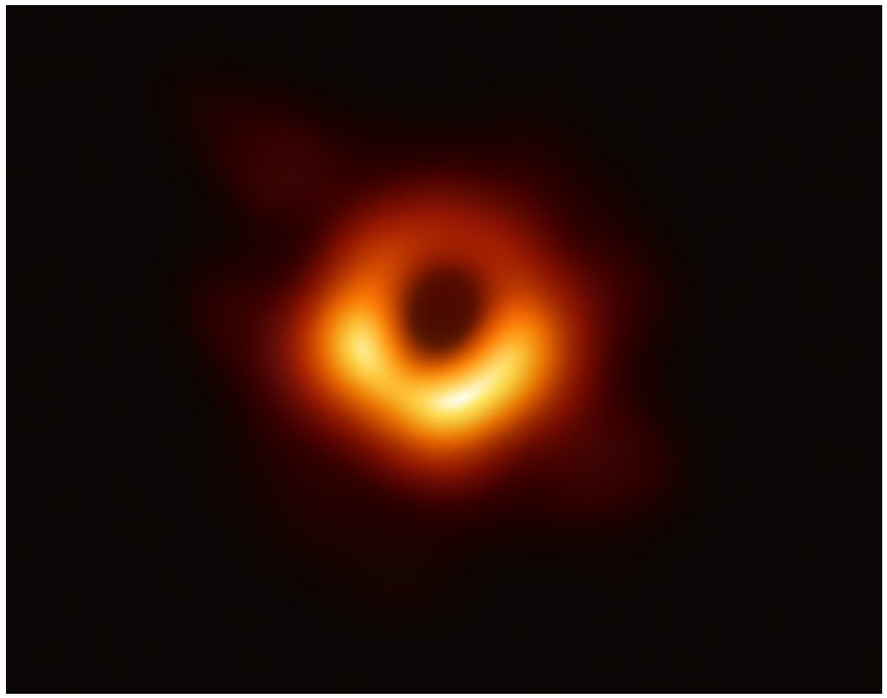
CSC4010: Introduction to Theoretical Computer Science



Lecture 18

Space Complexity

Computational Resources: 1) Time Complexity
2) Space / Memory
3) Randomness.





Left: The first reconstructed image of a black hole emerge from the Event Horizon Telescope data.

Right: Dr. Katie Bouman, imaging scientist, posing near hard drives containing some of the **petabytes of black hole imaging data** collected by telescopes around the world.

What can we do in Sublinear space?

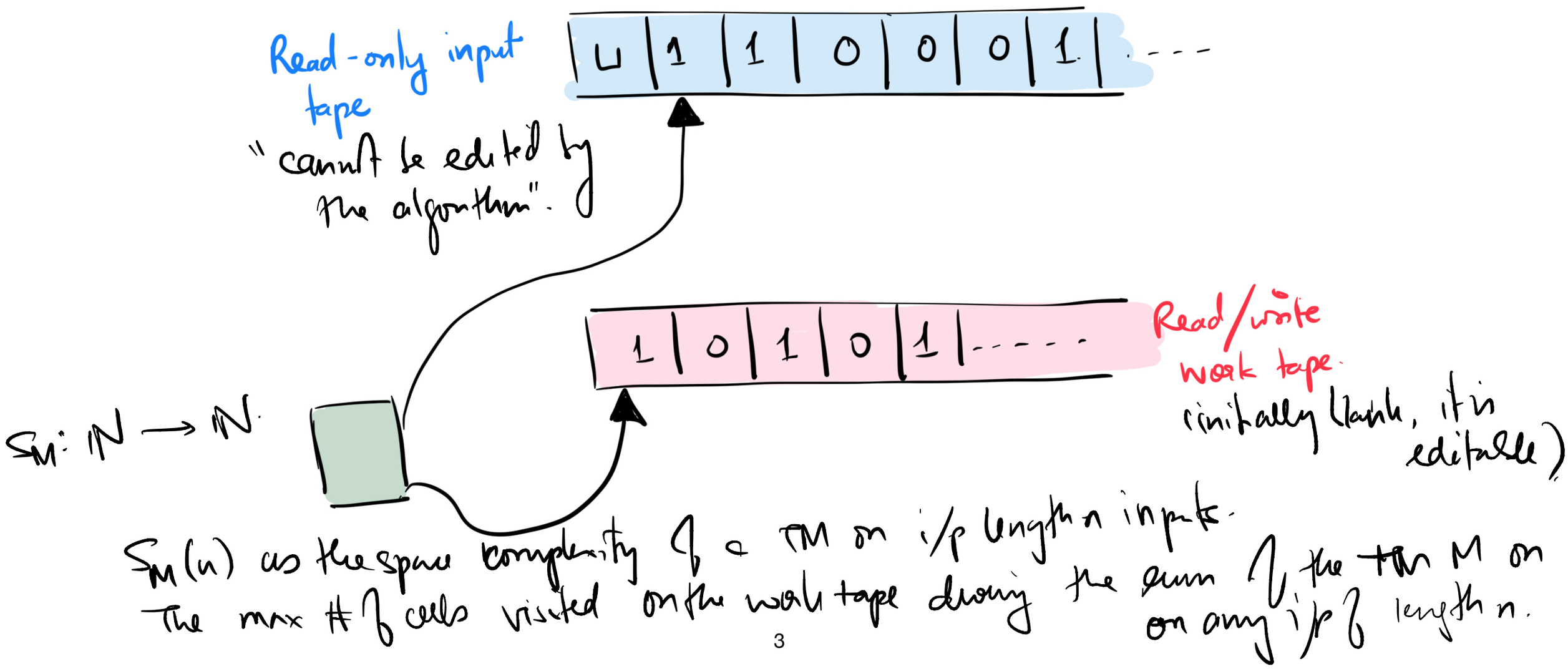
- The capacity of a device's readily accessible memory, such as RAM, is only a few gigabytes!

Space Complexity

sublinear space.

Can we solve any interesting problem in $o(n)$ space?

The one-tape Turing machine is not the right model of computation for studying sub linear-space algorithms. ⇒ The input like of the $\Omega(n)$ space.



$S_M(n)$ as the space complexity of a TM on i/p length n inputs.
The max # of cells visited on the work tape during the run of the TM M on on any i/p of length n.

Space Complexity: Complexity class SPACE(s)

Let L be a language and let $s : \mathbb{N} \to \mathbb{N}$ be a (space bound)

⟹ L is decided by a 2-tape TM that visits only $O(s)$ cells on the work tape.

Definition: $L \in \text{SPACE}(s)$ if there is a two-tape Turing machine M such that:

- M decides L — M halts on all i/p and $M(x)=1 \Leftrightarrow x \in L$.
- M never modifies the symbols written on tape 1
- We have $s_M(n) = O(s(n))$, where $s_M(n)$ is the maximum i such that the tape 2 head visits cell i during the computation of M on w for some $w \in \{0,1\}^n$. = the # of cells visited during the computation of M on length n i/p is $\leq O(s(n))$.

SPACE(s) = $\{L \subseteq \{0,1\}^*$ that can be decided by a TM with worst-case space $O(s)\}$

"Space" as a resource is very different from "time".

Complexity Classes

$\text{PSPACE} = \bigcup_{k \geq 1} \text{SPACE}(n^k)$

the class of all languages decidable by 2-tape TMs that take polynomial space.

$\text{L} = \text{SPACE}(\log n)$ (log space)

the class of all languages that can be decided by 2 tape TMs that take log space.

L is the set of languages that can be decided in logarithmic space.

$\text{EXPSPACE} = \bigcup_{k > 1} \text{SPACE}(2^{n^k})$

BALANCED $\in L$

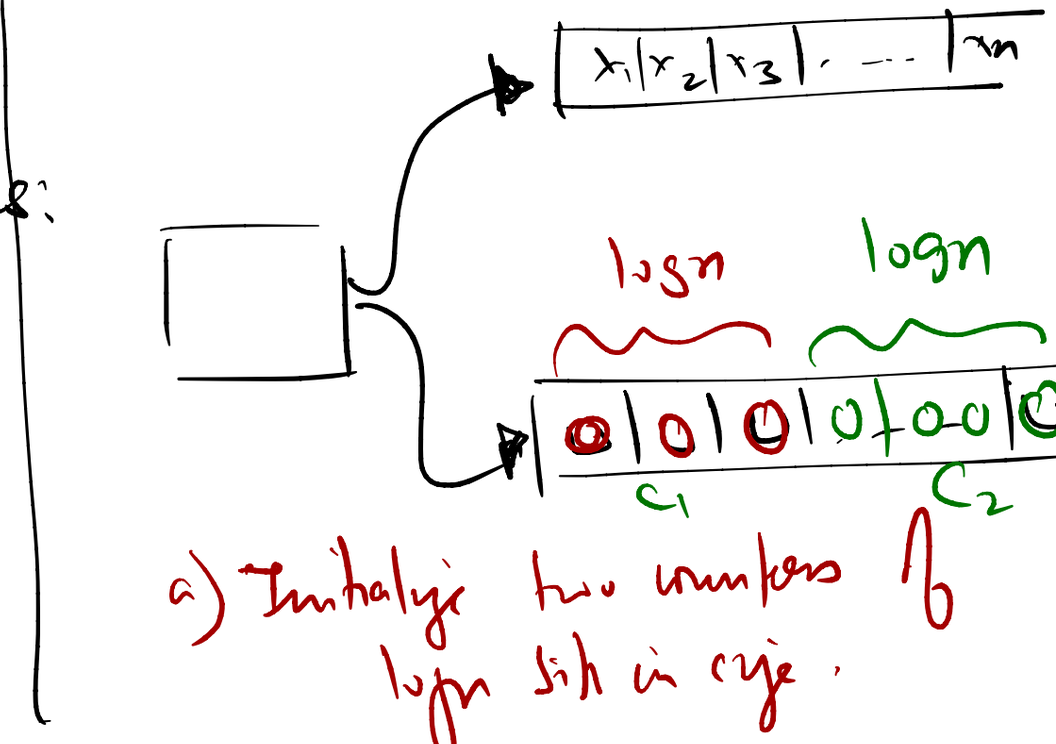
BALANCED = $\{x \in \{0,1\}^* : x$ has equal number of zeroes and ones$\}$

Claim: BALANCED $\in L$

Proof Sketch: Given $x \in \{0,1\}^n$:

Parse the i/p string, and increment the counters:

- Count the number of ones in x (incrementy C_2 when $x_i = 1$)
- Count the number of zeroes in x (increment C_1 when $x_i = 0$.
- Check if the two counts are equal



The total space taken $\leq 2\log n$.

BALANCED $\in$ SPACE$(\log n)$

$\Rightarrow$ BALANCED $\in L$.

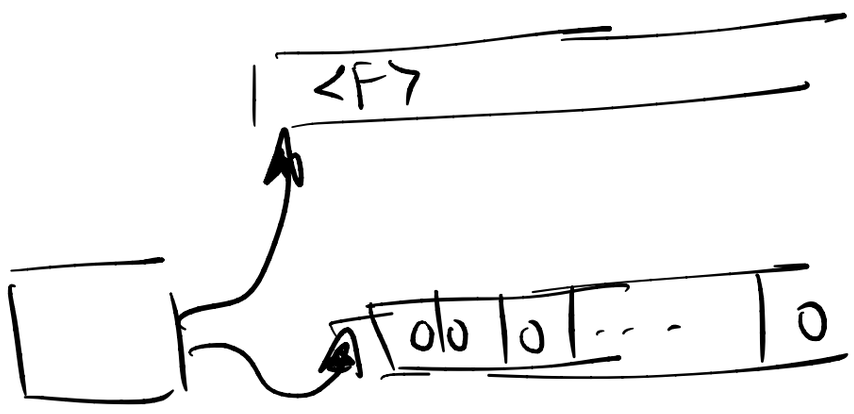
SAT ∈ PSPACE

Corollary: NP ⊆ PSPACE

SAT = { ⟨F⟩ | F is a satisfiable boolean formula }.

Suppose F is on n-input variables then, it satisfiable
⟺ ∃ x ∈ {0,1}^n s.t. F(x) = 1.

1)



a) Evaluate F on all settings of $x_1 = v_1, x_2 = v_2, \ldots, x_n = v_n$ where $v_i \in \{0,1\}$.

b) Evaluate F on $x_1 = v_1, x_2 = v_2, \ldots, x_n = v_n$ takes at most time $O(n^c)$

c) This implies $O(n^c)$ space to evaluate

d) Check if $F(v_1 v_2 v_3 \ldots v_n) = 1$
(i) If yes, then accept.
(ii) Else erase work tape & evaluate on next setting of variables.

SAT ∈ PSPACE

Corollary: NP ⊆ PSPACE

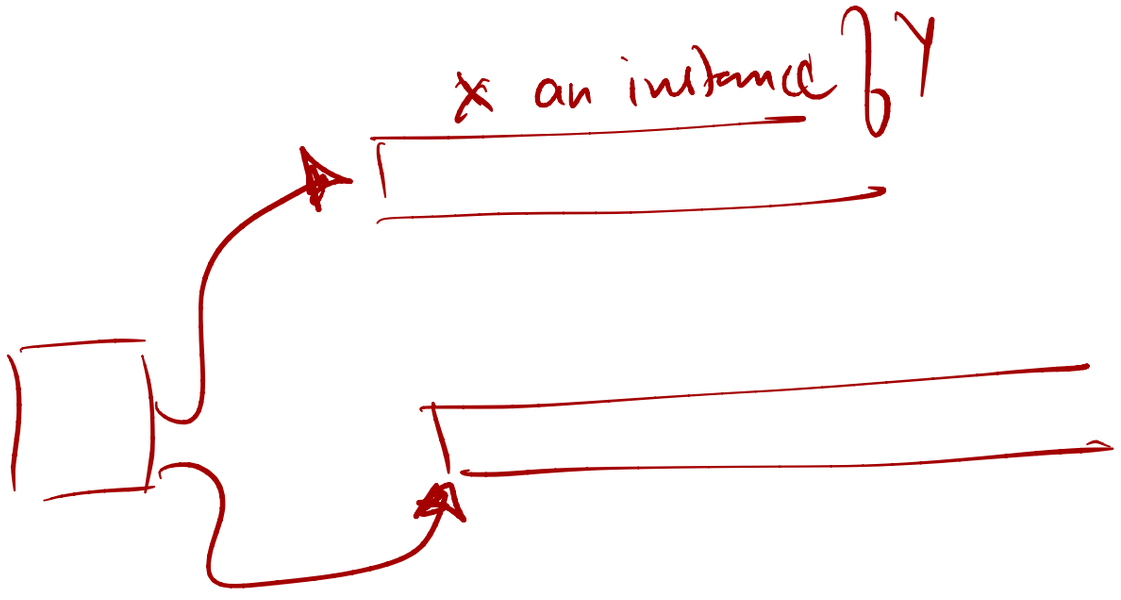
NP ≤ PSPACE

a) SAT is NP-hard ⇒ any $Y \in NP$, $Y \leq_P SAT$.

⇓

(If that polynomial time computable, and $x \in Y \Leftrightarrow f(x) \in SAT$)

b)



Algorithm:

(i) given x, compute f(x) and write it on tape 2.

(ii) Run the PSPACE algorithm for SAT on f(x).

Space complexity: a) $|f(x)| \leq O(n^c)$ for some $c \geq 0$ (as it polynomial time computable)

b) space complexity of the algorithm $\leq |f(x)|^k = O(n^{ck})$

1) $S(n) = O(T(n))$
2) $S(n) = \Omega(\log T(n))$

$\log T(n) \leq S(n) \leq T(n)$

↳ if we take $T(n)$ time, then, we can write more than $T(n)$ cells.

Time-Space Hierarchy

Theorem: For every function $s : \mathbb{N} \to \mathbb{N}$, $\text{SPACE}(s) \subseteq \text{TIME}(2^{O(s)})$

Any language L that can be decided in space $O(s)$, has a TM that decides it in time $\leq 2^{O(s)}$.

(i) Suppose M is the $O(s)$ space decider for L.
Consider any input x for M.
Then, we can consider the computational history of M.

$C_1 C_2 \dots \to C_T$

T is the time complexity of M.

$|C_i|$ is a string of length $\leq O(s)$ on $\Sigma \cup Q \cup \{\vdash\}$

(ii) As M halts, it cannot be the case that $C_i = C_j$ for some $i \neq j$.
(as that would mean that M is looping.

(iv) the total # of distinct configurations of length $s(n) \leq \gamma^{O(s(n))}$

Time-Space Hierarchy

Theorem: For every function $s : \mathbb{N} \to \mathbb{N}$, SPACE($s$) $\subseteq$ TIME($2^{O(s)}$)

(V) In the worst case M visits all possible configurations in comp. path.

$$\leq s^{O(s)}$$

$$\leq 2^{O(s)}$$

s is fixed.

$\Rightarrow$ Any language L, which can be decided in space $\leq s$ can be decided in time $\leq 2^{O(s)}$.

$$\text{SPACE}(s) \subseteq \text{TIME}\left(2^{O(s)}\right).$$

$C_1 \; C_2 \ldots C_3 \cdots C_j \; C_4 \; C_5 \ldots$

Space Hierarchy Theorem

Theorem: Let s be a space-constructible function. Then, $\mathrm{SPACE}(s) \subsetneq \mathrm{SPACE}(s \log s)$.

There are languages which can be decided in space $s \log s$, but cannot be decided in space s.

Space constructible function: s is space constructible if $\exists$ a TM that takes as i/p 1^n and outputs $s(n)$ in space $O(s(n))$

The # of cells visited during the computation is $O(s(n))$

Examples: $\log n$, n^c for $c \geq 1$, 2^{n^c} .....

Space Hierarchy Theorem

Theorem: Let s be a space-constructible function. Then, $\text{SPACE}(s) \subsetneq \text{SPACE}(s \log s)$.

Diagonalization: $Y = \{ \langle M \rangle \mid M \text{ rejects } \langle M \rangle \text{ in space } s \log s \text{ and time } \leq 2^{s \log s} \}$

(i) $Y \in \text{SPACE}(s \log s)$

(ii) $Y \notin \text{SPACE}(s)$

↳ The decider for Y will take as i/p $\langle M \rangle$, and simulate M on $\langle M \rangle$.

↳ Accept if M rejects $\langle M \rangle$ in $s \log s$ space & time $\leq 2^{s \log s}$

↳ Reject if: (i) M accepts $\langle M \rangle$ - - - -

(ii) M tries to take more space than $s \log s$ or does not halt in $2^{s \log s}$ steps

Space Hierarchy Theorem

Theorem: Let s be a space-constructible function. Then, SPACE(s) $\subsetneq$ SPACE($s \log s$) .

Diagonalization: $Y = \{ \langle M \rangle \mid M \text{ rejects } \langle M \rangle \text{ in space } s\log s \text{ and time } \leq 2^{s\log s} \}$

- (i) $Y \in$ SPACE($s\log s$)
- (ii) $Y \notin$ SPACE(s)

goal: $|\langle M_y \rangle| = n$, $k(n) > s(n)\log s(n)$.

M_y is a decider for Y and it has space complexity k

(i) $M_y(\langle M \rangle)$ accepts $\Leftrightarrow$ $M(\langle M \rangle)$ rejects in space $s\log s$ and time $\leq 2^{s\log s}$

(ii) Run $\langle M_y \rangle$ on M_y :

(a) M_y rejects $\langle M_y \rangle$ in space $\leq s\log s$ & time $\leq 2^{s\log s}$ $\Rightarrow$ M_y accepts $\langle M_y \rangle$.

(b) M_y accepts $\langle M_y \rangle$ in space $\leq s\log s$ $\Rightarrow$ M_y will reject $\langle M_y \rangle$.

$s \geq \Omega(\log T)$ → $s \geq f(s)\log(T)$

space $\leq s\log s$ but $> 2^{s\log s}$

(By time-space hierarchy)

space $= \Omega(s\log s)$

Space Hierarchy Theorem

Theorem: Let s be a space-constructible function. Then, $\text{SPACE}(s) \subsetneq \text{SPACE}(s \log s)$.

M_y does not halt on $\langle M_y \rangle$ in space $s\log s$ and $\leq 2^{s\log s}$.

(i) At least on $\langle M_y \rangle$ the space complexity $K(m) > \Omega(m)\log(c(m)) \cdot f(n)$

$\Rightarrow$ Y cannot be decided in space(s).

(Partial proof: to complete this for $\forall n \geq m$, add dummy states to M_y and repeat the diagonalization argument)

L vs. P vs. NP vs. PSPACE

L = SPACE(log n)

$L \subseteq P \subseteq NP \subseteq PSPACE$

What we expect: **All** of these containments are **strict.**

What we can prove: **At least one** of these containments is **strict.**

Theorem: $L \neq PSPACE$

↳ $P \subseteq NP$ (Easy to decide → easy to verify)

↳ $NP \subseteq PSPACE$ (SAT is NP-complete → SAT ∈ PSPACE)

↳ $L \subseteq P \Rightarrow$ $SPACE(S) \subseteq TIME(2^{O(S)})$

$L \subseteq TIME(2^{O(\log n)}) \subseteq P.$

$L \subsetneq PSPACE$ (Space hierarchy thm).

The L vs. P problem

We expect that $L \neq P$, but we don't know how to prove it.

$L = P$ would mean that every efficient algorithm can be modified so that it uses a tiny amount of workspace.

Class NL

NSPACE(s) stands for non-deterministic space s .

NSPACE(s) is the class of languages for which membership can be verified in space $o(s)$ and the verifier can read the certificate one time from left to right.

NL stands for non-deterministic logspace.

This is the class of languages for which membership can be verified in logarithmic space - with the additional requirement that the verifier can only read the certificate one time from left to right.

STCONN $\in$ SPACE$(k \log n)$

STCONN $= \{\langle G, s, t, k\rangle : G$ is a directed graph, there is a path of length $\leq k$ from s to t in $G\}$

STCONN $\in$ NL

STCONN $= \{\langle G, s, t, k\rangle : G$ is a directed graph, there is a path of length $\leq k$ from s to t in $G\}$

Theorem: STCONN $\in$ NL

Next Time

- There is a TM that decides STCONN in space $O(\log k \log n)$.

- **Savitch's Theorem:** For every $s : \mathbb{N} \to \mathbb{N}$, NSPACE$(s(n)) \subseteq$ SPACE$(s(n)^2)$

- PSPACE = NPSPACE