Lecture 19

What we have covered so far

1/3 decidable / ~~undedicidab~~ undecidable

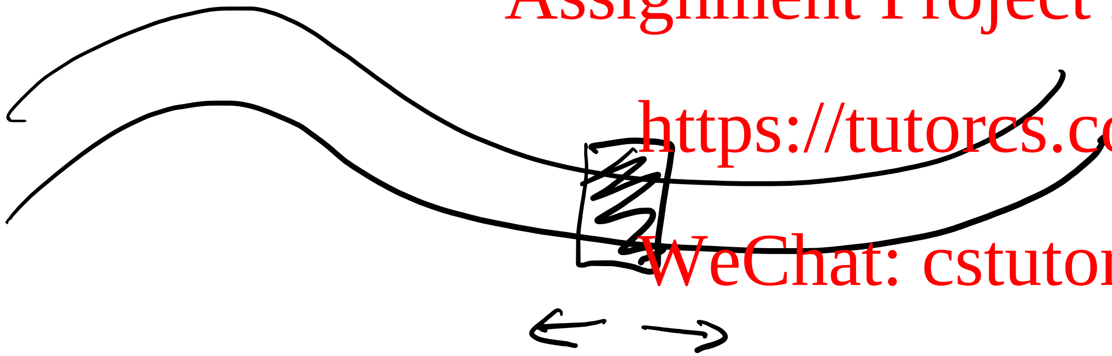
1/3 time complexity ~~⇒~~ times on T.M.

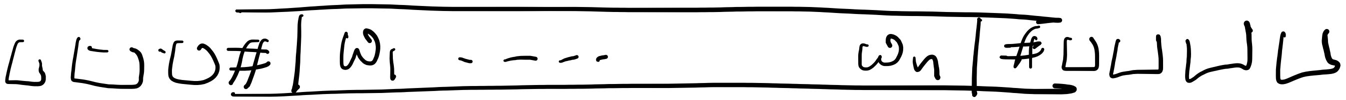
↳ NP, etc . . . , NP-completeness

NP ∩ coNP. → intermediate.

New measure of complexity (Space)

- Maximum space (number of cells) used by the Turing Machine





DSPACE

$L \in DSPACE(S(n))$

if $\exists$ Deterministic T.M. M which decides L using $\leq O(S(n))$ space where the length of the input is n.

$L = 0^n 1^n$.

$L \in DSPACE(n)$

NSPACE

$L \in NSPACE(S(n))$ if $\exists$ N.T.M. N which decides

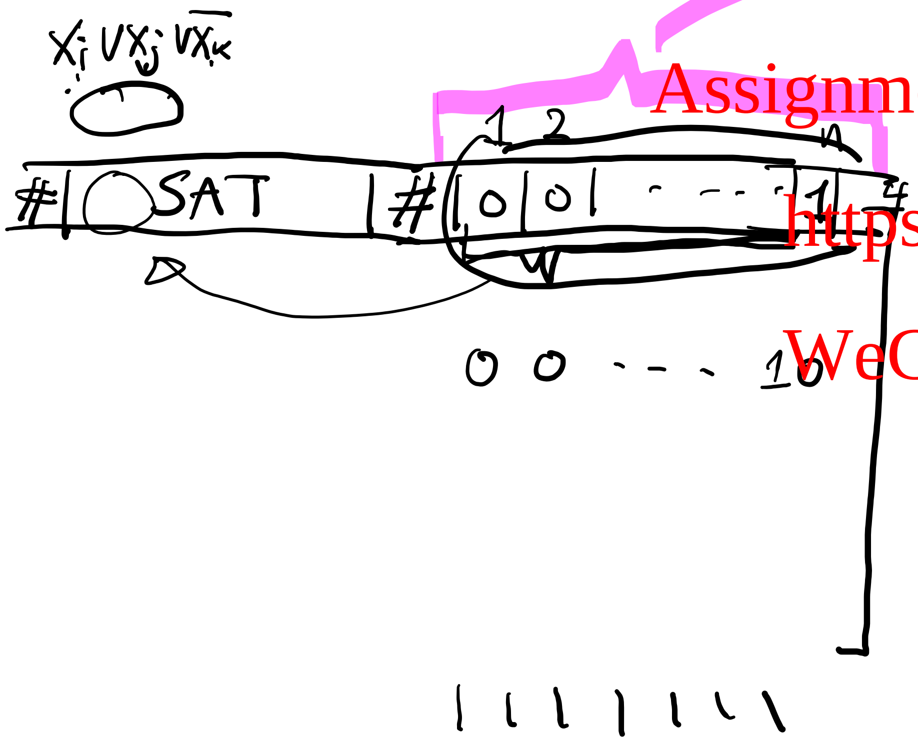
L using $O(S(n))$ space.

ex) 3SAT with n variables, $3n$ clauses.

easily check that $3SAT \in NSPACE(S(n))$

Ex) SAT

- SAT is in DSPACE (n)



<u>Idea</u>

Write a "current" assignment using $O(n)$ - space

check if "current" assignment satisfies the given SAT clause

if YES, Output Yes

if NO, move on to the next possible assignments.

Space is reusable! (while time is not)

First Theorem

- DTIME(S(n)) in DSPACE(S(n)) in NSPACE(S(n)) in DTIME(2^O(S(n)))

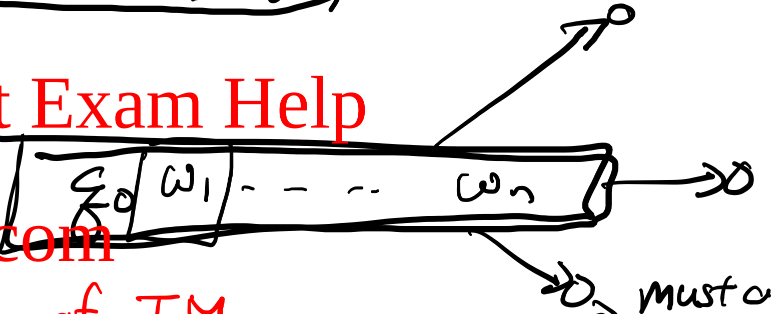
$$DTIME(S(n)) \subseteq DSPACE(S(n)) \subseteq NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$$

definition.

Proof of Cook-Levin.

* Configuration Graph of TM (Directed)

- Vertices: → "snapshot" of each T.M.
- Edges :



$$(\underbrace{|\Sigma|}_{c_1} + \underbrace{|Q|}_{c_2})^{O(s(n))}$$

by the definition of T.M.

must also be some constant

$$= \left(2^{\log(c_1+c_2)}\right)^{O(s(n))} = 2^{O((\log(c_1+c_2))\cdot s(n))}$$

$$= 2^{O(s(n))}$$

$(v_1, v_2) \in E$

if v_1 yields v_2

Properties

- Description of each nodes

by T.M. configuration

V_1 V_2

- Edges can be described by a CNF formula i.e. $\phi_{M,x}(C, C') = 1$ if and only if C and C' are neighbors – $\boxed{\text{Cook-Levin Thm}}$

initial input

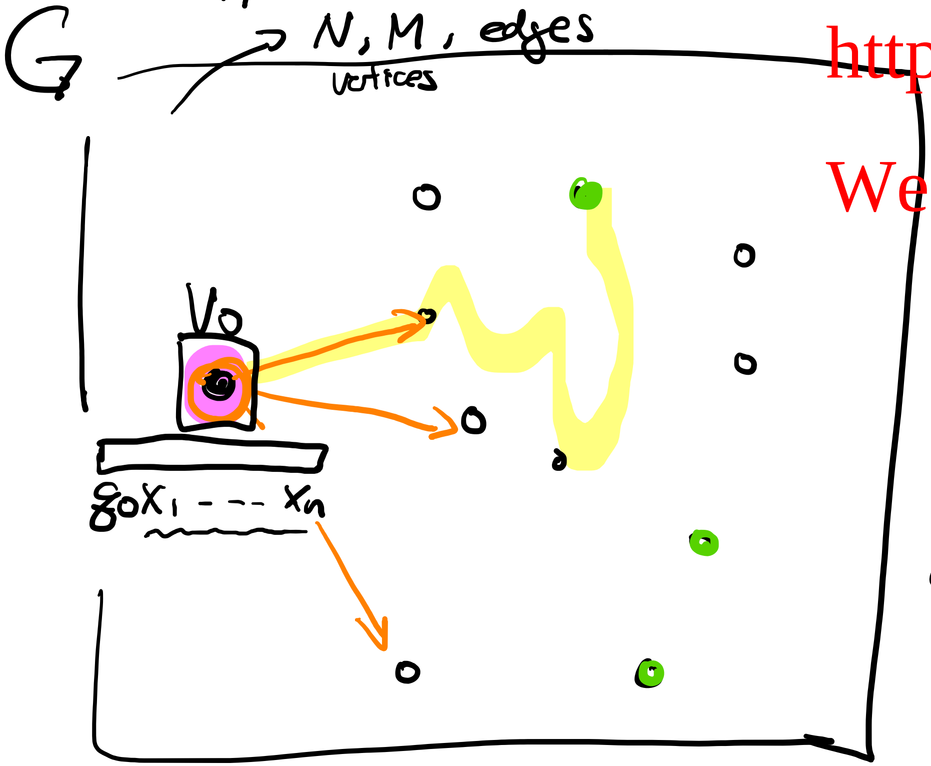
(N).T.M

NSPACE(S(n)) in DTIME(2^O(S(n)))

Start $S = \{v_0\}$.
iteratively, add neighbors to S
Halt if S stays the same.
Check if $S \cap$ ● $= \phi$ or not.

- Connectivity between accepting config and starting config using standard connectivity algorithm

Suppose $L \in NSPACE(S(n))$, $\Rightarrow \exists$ N.T.M. N which decides L using $O(S(n))$-space.



$O(poly(N))$ time where N # of vertices.

# of vertices $\le 2^{O(S(n))}$

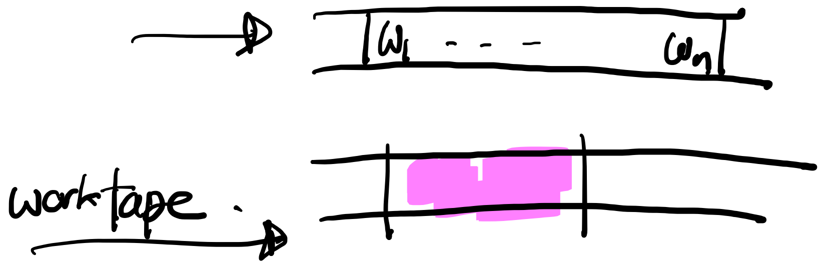
● → configs with g_{acc}.

$x \in L$ iff $\exists$ a path from V_0 to one of the ● node.

start with V_0, then ask if ● is reachable from V_0 $\Rightarrow L \in DTIME(2^{O(S(n))})$

$\in O(N^c) = O((2^{O(S(n))})^c) = 2^{O(S(n))}$.

Our Goal till the finals



- Savitch's Theorem

$$NSPACE(S(n)) \subseteq DSPACE(S(n)^2)$$

- PSPACE Completeness (?)

$\forall\ L \in PSPACE$

$$PSPACE = \bigcup_{c>0} DSPACE(n^c) = \bigcup_{c>0} NSPACE(n^c)$$

$L \leq_p L' \rightarrow L'$ as PSPACE-hard.

$L' \in PSPACE$,

L' is PSPACE-complete.

$NP \stackrel{?}{=} coNP$.

- LogSpace, NL = coNL

$L \subseteq P$.

$NL \subseteq P$.

$L =$

$NL = coNL$.

Relation between NSPACE and DSPACE

Clearly

- DSPACE contained in NSPACE (with same function);
- Question) NSPACE(S(n)) contained in DSPACE(?)

$$DSPACE(S(n)) \subseteq NSPACE(S(n))$$

$$NSPACE(S(n)) \subseteq DSPACE(\,?\,).$$

$$NSPACE(S(n)) \subseteq DTIME\left(2^{O(S(n))}\right) \subseteq DSPACE\left(2^{O(S(n))}\right).$$

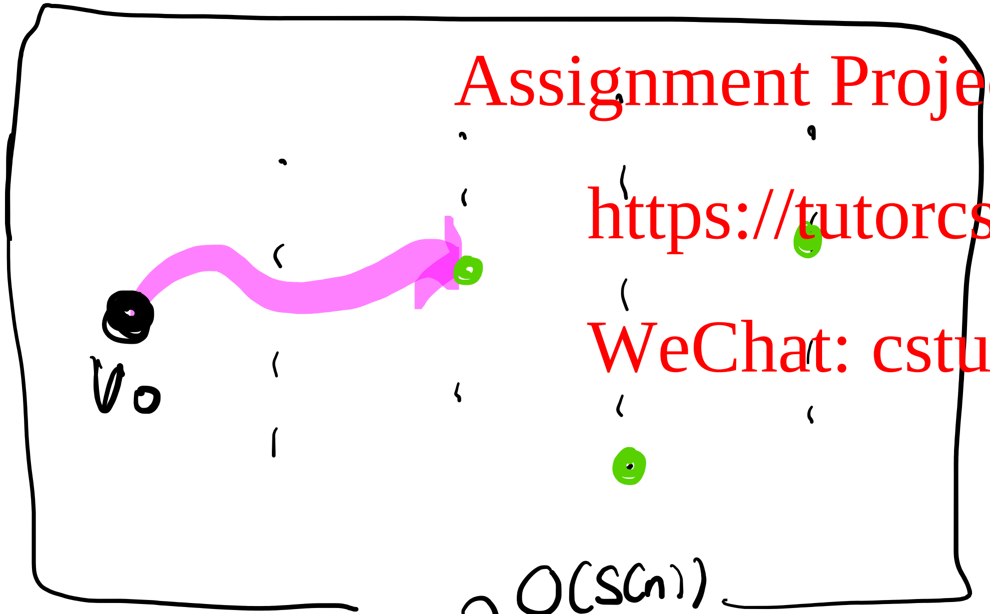
Savitch’s Theorem

- There is only polynomial blowup between DSPACE and NSPACE
- NSPACE(f(n)) is in DSPACE(f^2(n))

First attempt on Proof

$N., x$

- Just enumerate all possible paths starting from **starting configuration**



$2^{O(S(n))}$

check if $\exists$ path which ends in

- **TOO COSTLY…**

you must store the "current" path $\rightarrow$ $2^{O(S(n))}$ space.

Proof

- Recursively ask R(u,v,i); if you can reach from u to v in 2^i steps.

- True iff there exists z such that R(u,z,i-1) and R(z,v,i-1) both true.

Pictorial Description

So given table for i-1, how much extra space?

- S(i) = S(i-1) + O(log M) where M is the # of vertices – why ?

- Then S(log M) = O(log^2 M)

PSPACE