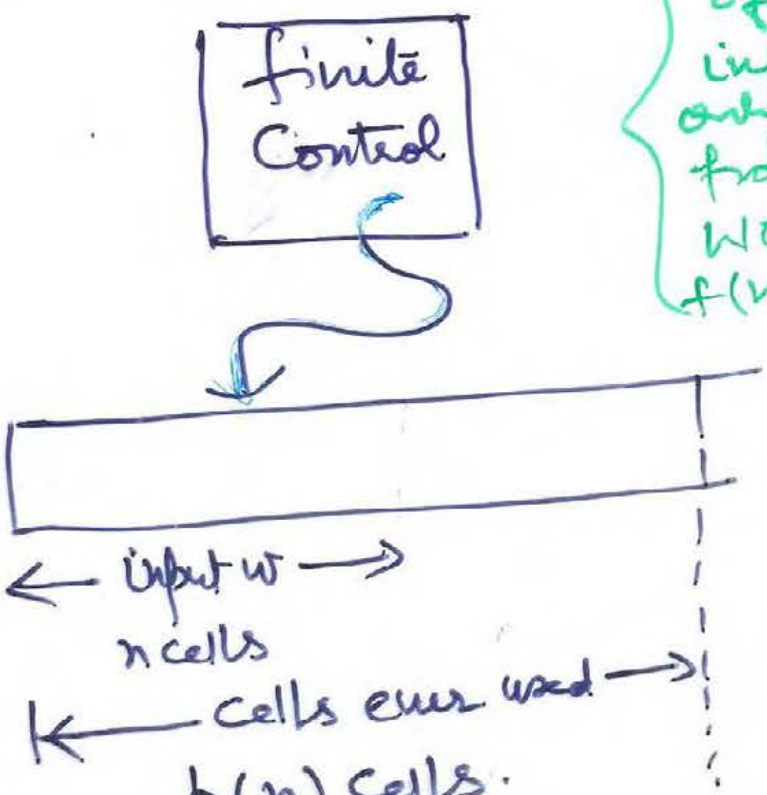
Space Complexity

→ A Turing M/C runs in space $f(n)$ if it scans a maximum of $f(n)$ tape cells for any input of length n

Modification in the definition of TM:
input is read only and separate from the read/write WORK Tape.
$f(n)$: No. of tape cells on the work tape



→ SAT is $O(n)$ space

input ϕ with $x_1 \cdots x_k$.

1. create array T, mapping variable indices. $1 \cdots k$ to true or false
2. for every possible value of T:
   (i) Copy ϕ to work tape
   (ii) Replace the variables of ϕ with the Constants given by T
   (iii) Calculate formula
   (iv) if result == true accept ϕ
3. Reject ϕ.

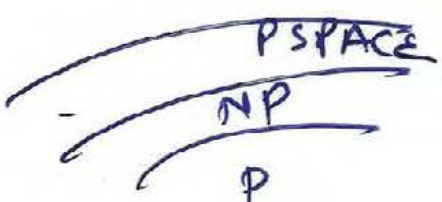
T: $O(n)$ space
scratch copy of ϕ = $O(n)$ space

Example: DBL – Word :

1. If w is of odd length, reject
2. Compute $m = |w|/2$
3. for i from 1 to m do the following:
   (i) Record the current tape symbol in l
   (ii) Move right by m cells
   (iii) if the current tape cell ≠ l reject
   (iv) Move left by (m-1) cells
4. Accept.

How much work space does this require?

m : O(log n) space
l : O(1) space
Total : O(log n) space



Polynomial space

SPACE $(f(n)) = \{L : L$ is decided by an O(f(n)) space deterministic Turing m/c$\}$

NSPACE$(f(n)) = \{L : L$ is decided by an O(f(n)) space non-deterministic TM$\}$

$$PSPACE = \bigcup_k SPACE(n^k)$$

$$NPSPACE = \bigcup_k NSPACE(n^k).$$

Note:

(1) A Turing Machine M that uses space $f(n)$ runs in time $O(f(n))2^{O(f(n))}$

$\therefore$ $f(n)$ choices for tape head position and $2^{O(f(n))}$ tape contents

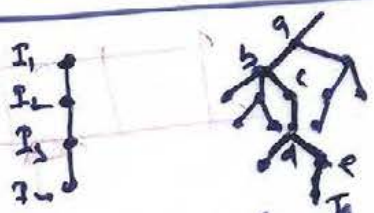
(2) A Turing M/C M that runs in time $f(n)$ uses $O(f(n))$

$P \subseteq PSPACE$

$NP \subseteq NPSPACE$

THM: if M is a polynomial-space bound TM (deterministic or Non-deterministic) and $p(n)$ is it's polynomial space bound then there is a constant c such that if M accepts its input w of length n, it does so within $c^{1+p(n)}$ moves.

Def



proof outline:

t = No. of tape symbols

s = No. of states.

No. of different ID's when only $p(n)$ tape cells are used is at most $s\,p(n)\,t^{p(n)}$.

$\Rightarrow c = s + t^{(1+p(n))}$

$(t+s)^{1+p(n)} = t^{(1+p(n))} + (1+p(n))s\,t^{p(n)} + \cdots$

Savitch theorem:

for any f where $f(n) \geq n$ $\forall n$

$$NSPACE(f(n)) \subseteq SPACE(f^2(n))$$

proof outline:

Consider the deterministic test for whether a NTM N can move from ID I to ID J in at most m moves.

→ A DTM D systematically tries all middle ID's K to check whether I can become K in $m/2$ moves and then K can become J in $m/2$ moves

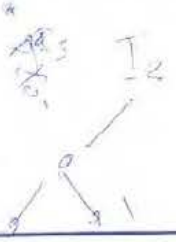
```
Boolean function reach(I, J, m).
  Begin
      if (m==1) then
        test if I==J or I can become J
          after one move
        return true if so, false if not.
      end;
      else
        for each possible ID K DO
        if (reach(I, K, m/2) AND reach
                              (K, J, m/2)) then
          return true;
        return false
      End;
  end;
```

Tape of a DTM simulating a NTM by recursive calls to reach.

$I_1 J_1 m$	$I_2 J_2 m/2$	$I_3 J_3 m/4$	$I_4 J_4 m$	8	...

$\log_2 m$.

$m \leq c^{p(n)}$



$O(p(n))$.

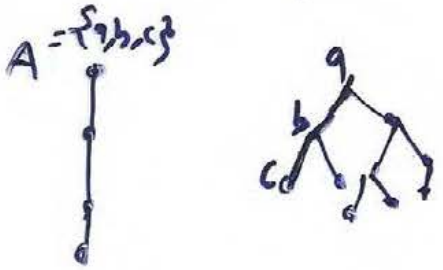
→ Each stack frame themselves take $O(p(n))$ space.

→ PSPACE Completeness

Defn: A problem B is P-SPACE Complete if

(i) $B \in PSPACE$ and.

(ii) for every $A \in PSPACE$, $A \leq_p B$.



Quantifiers and games.

Proof :-

TQBF = {ϕ : ϕ is a true fully Quantified Boolean formula} is PSPACE COMPLETE

1. if ϕ is quantifier free
   - 1.1 for ∀ possible truth assignment.
     - 1.1.1 if the assignment satisfies. ϕ accept.
   - 1.2 reject.
2. Convert ϕ into an equivalent formula ψ in PNF.
3. if $\phi = \exists x \psi$ for some ψ.
   - 3.1 if $\psi[0/x]$ is in TQBF, accept.
   - 3.2 if $\psi[1/x]$ is in TQBF accept else reject.
4. if $\phi = \forall x \psi$ for some ψ.
   - 4.1 if $\psi[0/x]$ is not in TQBF reject.
   - 4.2 if $\psi[1/x]$ is not in TQBF reject else accept.

Note: $\forall x \exists y [y > x]$ and $\exists x \forall y [y > x]$ are different

when each variable of a formula appears within the scope of some quantifier it's called fully Quantified

every problem $A \in PSPACE \leq_P TQBF$.
if $A \in PSPACE$, it is decided by a Turing M/C that runs in space $n^k = f(n)$.

WE WOULD LIKE TO MAP EACH $w \in A$ to a formula ϕ THAT IS TRUE IF AND ONLY IF M_1 ACCEPTS w.

First idea:

ϕ a $n(h)$

h = no. of steps that M_1 like

~~· or longest ϕ [illegible]~~

use $\phi_{c_1, c_2, t}$: true if M_1 can go from c_1 to c_2 within t steps

let $\phi = \phi_{C_{start}\, C_{accept}\, h}$ $\quad h = 2^{df(n)}$.

$$\phi_{c_1,c_2,t} = \begin{cases} (c_1 = c_2) \wedge (c_1 \text{ yield } c_2 \text{ in single step}) & t=1 \\ \exists m_1 (\phi_{c_1 m_1, t/2} \wedge \phi_{m_1, c_2, t/2}) & t>1 \end{cases}$$

but + doubles the length.

so use another Quantifier

$$\exists m_1 \forall c_3 \forall c_4 \left(c_3 = c_1 \wedge c_4 = m_1 \vee c_3 = m_1 \wedge c_4 = c_2\right)$$

$$\Rightarrow \phi_{c_3, c_4, t/2}$$

$O(f(n))$ length added at each step.

$\log\left(2^{df(n)}\right) = O(f(n))$ recursive steps

The No. of level of recursion $= \log\left(2^{df(n)}\right)$ or $O(f(n))$

Hence the size of the resulting formulae $O(f^2(n))$.

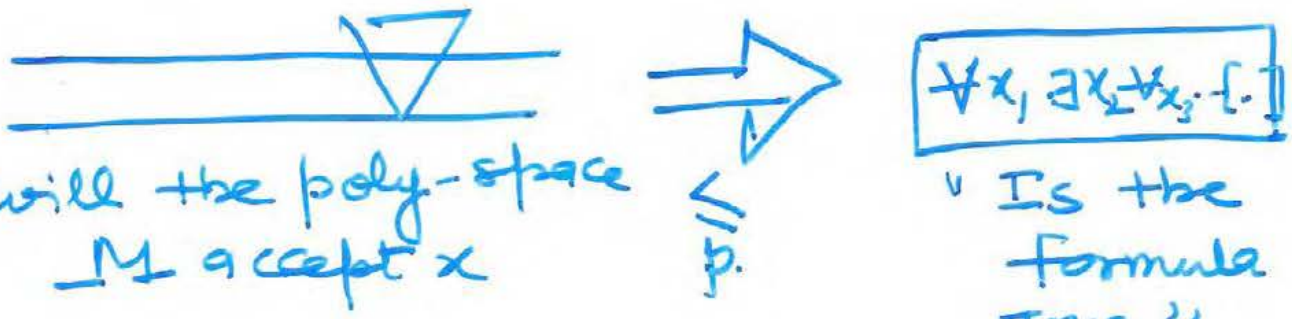
$$\phi_{c_1, c_2, t} = \exists m_1 \forall (c_3, c_4) \in \{(c_1, m_1), (m_1, c_2)\} \left[\phi_{c_3, c_4, \lceil t/2 \rceil}\right].$$

$\phi_{\text{Obtain}, c_a \text{ capt}, h}$ where $h = 2^{df(n)}$

ANOTHER proof

TQBF is PSPACE Complete

It remains to show TQBF is PSPACE-hard.



→ Given a Turing Machine M for a language L ∈ PSPACE and an input x.

Let $f_{M,x}(U,l)$

→ True iff M on I/p x moves from Configuration U to the Configuration l

Hence

Formulating Connectivity

The following formula, over variables $U, u \in V$ and path's length d, is true iff G has a path from u to v of length $\leq d$.

$\Phi(u,v,1) \equiv f_{m,x}(u,v) \vee u = v$

$\Phi(u,v,d) \equiv$

$\exists w \forall x \forall y [((x=u \wedge y=w) \vee (x=w \wedge y=v)) \rightarrow \Phi(x,y,d/2)]$

w is reachable from u in $\lceil d/2 \rceil$ steps. v is reachable from w in $\lceil d/2 \rceil$ steps

Simulatio AND of $\Phi(u,w,d/2)$ and $\Phi(w,v,d/2)$.

$\Phi \equiv \Phi(s,t,|v|)$ is true iff there is a path from s to t.

Φ is constructable in poly-time

Thus any PSPACE language is poly-time reducible to TQBF.

Quantifiers and games.

$\phi_1 \equiv$

$$\exists x_1 \forall x_2 \exists x_3 \left[(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}) \right]$$

→ Two players E and A.

→ E picks $x_1 = 1$ ✓
A picks $x_2 = 0$ ✓
E picks $x_3 = 1$.

ϕ is 1, player E won.

→ player E has a winning strategy for this game. A player has a winning strategy for a game if that player wins when both sides play optimally

→ Change the formula

$\phi_2 \equiv$

$$\exists x_1 \forall x_2 \exists x_3 \left[(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3}) \right]$$

player A now has a winning strategy because. No matter what player E selects for x_1

formula game.

1. Start with quantifier free formulae ψ with $2k$ variables
2. k rounds of game.
   - 2.1 Player T chooses value for x_{2k-1}
   - 2.2 Player F chooses value for x_{2k}.
3. if resulting truth assignment makes ψ true, T wins; else F wins.

Thm: FLAGAME is PSPACE Complete

FLAGAME $\in$ PSPACE

→ Given ψ, we construct $\exists x_1 \forall x_2 \cdots \exists x_{2k-1} \forall x_{2k} \psi$ and determine whether this is true:

→ TQBF $\leq_P$ FLAGAME

$\phi \longrightarrow$ Convert into PNF requires polynomial space.

$\phi' = Q_1 x_1 \cdots Q_k x_k \psi$

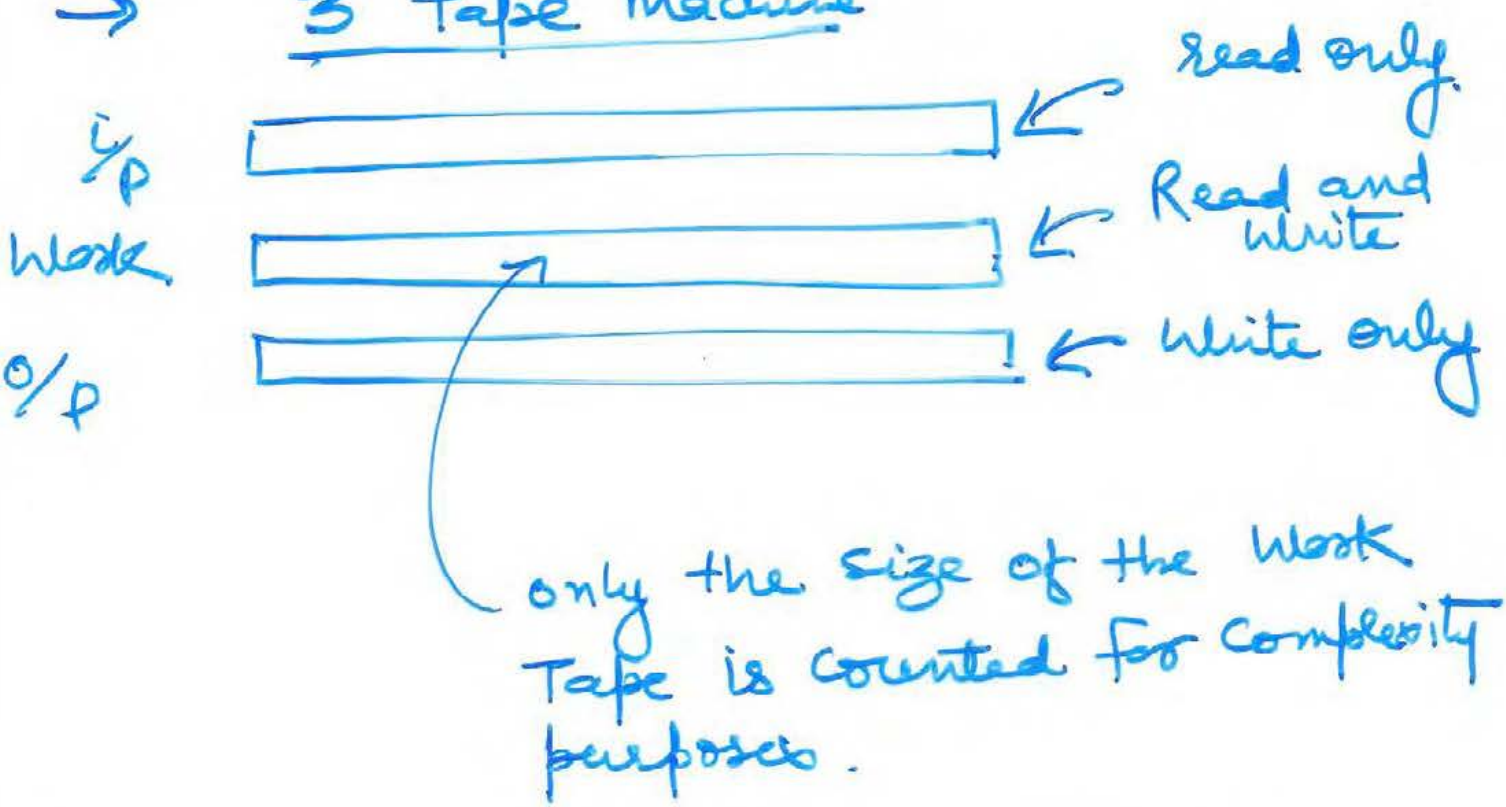
if $Q_1 = \forall$, append $\exists z_0$

$\exists$ $\forall z_0$.

$\phi'' \longrightarrow \phi$ Conversion process takes only $O(n)$

How can TM use only $\log n$ space if the input itself takes n cells?

→ 3 Tape Machine



Qn: How much space would a TM that decides $\{a^n b^n \mid n>0\}$ require?

Soln:- To count upto n, we need $\log n$ bits

Graph Connectivity

CONN :-

Instance: a directed graph $G=(V,E)$ and two vertices $s, t \in V$

Problem :- To decide if there is a path from s to t in G

CONN is in NL

- start at s
- for $i = 1, \ldots |n|$ {
  - Non-deterministically choose a neighbor and jump to it
  - Accept if you get to t.

}

Otherwise reject.

Note :-

- counting upto $|n|$ requires $\log |n|$ space
- storing the current position requires $\log |n|$ space

Logarithmic space

Note

$L = SPACE(\log n)$

$NL = NSPACE(\log n)$

→ A transducer is a Turing m/c with a read only input tape, a write only output tape, and a read/write work tape

Note → A function f is log space computable if there is a transducer that computes f using only $O(\log n)$ work space

Example:

$PATH = \{\langle G, s, t\rangle : \exists$ s to t in $G\} \in NL$

Non-deterministic decision algorithm for PATH

1. Initialize c to s
2. for i from 1 to n
   (i) if c = t accept
   (ii) Choose a successor d of c
      → if no such d exist, reject
      → Update c to d.
3. reject

M/c only stores a pointer to the current node and runs a maximum of n iterations where n is the No. of nodes

M/c uses $\log n$ space

where $n = 2^b$ and b is the No. of bits required to count upto n ≤ n

c_d_i_h : $O(\log n)$

Probabilistic Computation

→ A probabilistic Turing m/c M_1 is a type of non-deterministic TM where each non-deterministic step is called a coin-flip step and has two legal next moves.

→ probability to each branch b of M_1's computation on input w

$$Pr[b] = 2^{-k}.$$

k = No. of coin flip steps that occur on branch b.

$$Pr[M_1 \text{ accepts } w] = \sum_{b \text{ is an accepting branch}} Pr[b].$$

$$Pr[M_1 \text{ rejects } w] = 1 - Pr[M_1 \text{ accepts } w]$$

→ M_1 recognizes language A with error probability ϵ if $0 \le \epsilon < \frac{1}{2}$

1. $w \in A \Rightarrow Pr[M_1 \text{ accepts } w] \ge 1 - \epsilon$
2. $w \notin A \Rightarrow Pr[M_1 \text{ rejects } w] \ge 1 - \epsilon$.

→ BPP is complexity class associated with probabilistic computation

In relativization method -

TM could use the oracle to solve any NP problem in polynomial time. regardless of P = NP, because. every NP problem is polynomial time reducible to the satisfiability problem. such a turing m/c is said to be computing relative to the satisfiability problem.

(relativization).

→ oracle allows the TM to test membership in the language without actually having to compute one. answer itself.

→ An oracle is a language A

→ An oracle Turing m/c M^A is an ordinary Turing m/c with an extra tape. Whenever M writes a string on the oracle tape it is informed whether the string is a member of A in a single step.

(A)

(B)

Interactive Proof Systems

- Languages in NP are those whose members all have short certificate of membership, which can be easily verified.

- {Proof Systems}

  - There is a mighty powerful prover
  - Prover needs to convince a verifier that the i/p indeed a member of language
  - so it sends the verifier a short (polynomial) certificate
  - The verifier has limited resources the verification of the certificate can not take more than polynomial time

Ex:- 3SAT

We would like to check the membership of a given formula

$$(x \vee y \vee z') \wedge (x' \vee y') \wedge z'$$

→ The prover must convince the verifier this formulae is satisfiable so it sends it an assignment which supposedly satisfies the formulae it's not difficult for the mighty prover to find such, if such exists

← Verifier simply needs to check the truth value of the formula under the assignment it received in order to ~~pass~~ find out whether prover was right, this normally takes polynomial time

Interactive proofs :-

- generalisation of the concept of a proof system
- obtained by adding two more features to the model
  - allowing a two-way dialog b/w the parties (interaction)
  - allowing the verifier to toss coins (randomness)
- An IPS for a language L is a two party game b/w a verifier and a prover that interact on a common input in a way satisfying following properties
  - * → verifier strategy is a probabilistic polynomial time

* → Correctness requirement

- Completeness : ∃ a prover strategy P :. ∀ x∈L when interacting on a common i/p x, P convinces verifier with probability of at least 2/3

- Soundness :- for $\forall x \notin L$, when interacting on the common i/p x, any prover strategy p^* Convinces the verifier with the probability of at most $1/3$

→ Complexity class IP consists of all the languages having an IPS

→ No. of messages exchanged during the protocol b/w two parties is called the number of rounds in the system

→ probabilistic analog of NP. can be defined by Interactive proof system.

→ NP = ~~Those~~ member have. short certificates of membership that can be checked easily.

OR

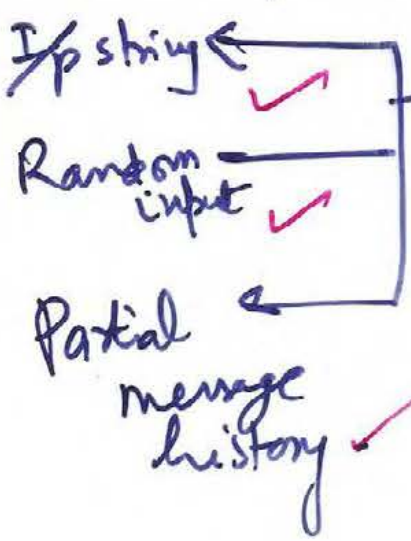
prover finds the proof of membership and a verifier checks them.

↓

{ polynomial time bounded.

→ if verifier is probabilistic polynomial time m/c.

→ IPS = prover + verifier + interaction



↓

Computes the next transmission to the prover from the message history sent so far.