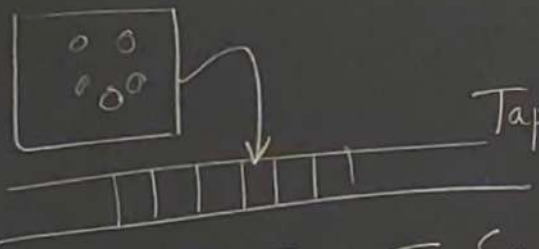
Computational Complexity Theory

Turing Machine. model of computation



Transition fn $\delta : \Sigma \times Q \to \Sigma \times \{L, R\}$

Alphabet — State space

TMs are incredibly robust.

Variants of TMs can be implemented by a "standard" TM

Standard TM. $\Sigma = \{0, 1, \sqcup\}$

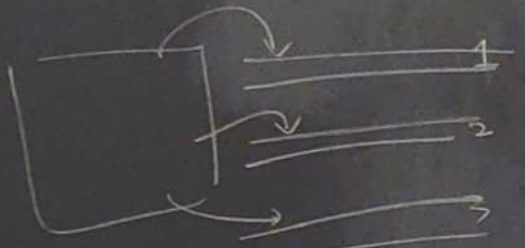
Single Tape

Head always moves left or right by single step.

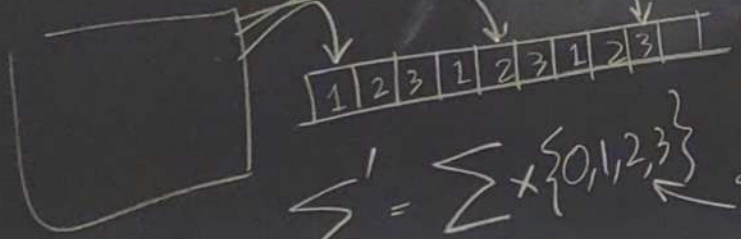
Variants of TMs

1) Larger Σ (Binary encoding)

2) More tapes



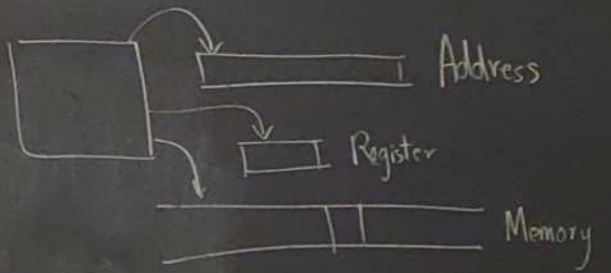
Interleave



$\Sigma' = \Sigma \times \{0,1,2,3\}$ ← Storing whether head is at position

get a single head

3) RAM model



Given an address, the TM fetches the content of address in memory into register.

Computational Complexity Theory

Def: Let $t(n): \mathbb{N} \to \mathbb{N}$ be a constructible/ function computable

$\mathrm{DTIME}(t(n))$ is the class of languages decidable in $t(n)$ time.

$\mathcal{L} \in \mathrm{DTIME}(t(n))$ if $\exists$ standard TM (and constant c) that decides $\mathcal{L}$ and runs in $\leq c \cdot t(n)$ time.

$\mathrm{DTIME}(t(n))$ is independent of alphabet.

$\mathrm{DTIME}_k(t(n))$ is for k-tape TMs

$$\mathrm{DTIME}_k(t(n)) \subseteq \mathrm{DTIME}(t(n) \log t(n))$$

Resources of interest:

1) Time: # steps a TM makes
2) Space. # tape cells accessed.

Thm: [Simulation theorem]

A k-tape TM that runs in $t(n)$ time can be simulated by a standard TM that runs in $c \cdot k \cdot t(n)^2$ time.

↖ constant

single head

Thm [Hennie-Stearns 71]
Efficient simulation in $ck\, t(n) \cdot \log t(n)$ time

Def. $P = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(n^k)$

P does not depend on specifics of TM

[Strong Church-Turing thesis, Cobham's thesis 62]

(1) All physically realizable models of computation can be simulated by standard TMs with polynomial overhead in run time.

P is the class of efficiently decidable languages.

(2) P has nice "closure" properties

"An efficient" algorithm can call an efficient algorithm.

(Polynomal calls to a polynomal time algorithm is still polynomial time.)

(3) [Edmonds 64, Gödel 52, Trakhenbrots 50's]

For many languages, algorithm in P implies beating brute-force.

Def: $\mathbb{P} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(n^k)$

$\mathbb{P}$ does not depend on specifics of TM

Def: $\mathbb{NP}$ $\mathcal{L} \in \mathbb{NP}$ if $\mathcal{L}$ is "efficiently verifiable".

$\exists$ polynomials p, t and a TM M

s.t. $\forall x, |x| \leq n$ $\in \{0,1\}^*$

(1) If $x \in \mathcal{L}$

$\exists$ "certificate" y, $|y| < p(n)$

and $M(x,y)$ accepts in $t(n)$ time.

(2) If $x \notin \mathcal{L}$

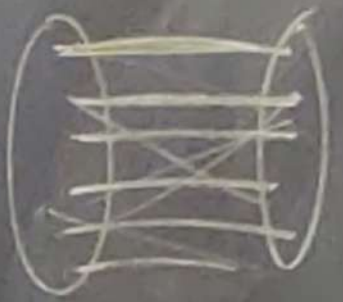
$\forall$ "certificates" y, $|y| \leq p(n)$

$M(x,y)$ rejects in $t(n)$ time.

Complexity Theory

$\mathcal{L} = \{ G = (U, V, E),$ G bipartite & G has perfect matching $\}$

$\mathcal{L} \in NP$



Q. Hilbert's 10th problem

Given a set of Diophantine equations, polynomial

is there an integer solution?

$x^2 + y^2 = z^2$

$2x + 3y = 10z$

Computational Comp

Q. A is integer matrix

b is vector

Solve $Ax = b$

n is the number of bits to write entries of A, b

(A is square, solution is given by Cramer's rule)

$$x_i = \frac{\det(\square\,\|\,b\,\|\,\square)}{\det(A)}$$

lexity Theory

Thm: $\mathbb{P} \subseteq \mathbb{NP}$

Verifier can ignore certificate.

Proof: Suppose $\mathcal{L} \in \mathbb{P}$. $\exists$ poly time TM M that decides $\mathcal{L}$.

The verifier M' does the following.

(1) Run $M(x)$ and follow output.