CPSC 421/501 Nov. 8, 2023

- CONNECTED-GRAPHS $\in$ P

$P = \{ L \mid L$ is decidable by a polynomial-time T.M.$\}$

Many examples in CPSC 320 of poly-time algorithms...

=

G graph $\rightarrow$ $G = (V, E)$

V is a set, $E \subset \{$pairs of vertices$\}$

e.g. $V = \{ a, b, c, \alpha, \beta, \sqrt{2}, \ldots \}$ ☹

To make sense of $\langle G \rangle$

we assume

$$V = \{1, \dots, N\}, \quad N \in \mathbb{N}$$

GR ...

$$V = \{v_1, v_2, \dots, v_N\}$$

identify

$$\begin{array}{cccc} v_1, & v_2, & \dots, & v_N \\ \updownarrow & \updownarrow & & \updownarrow \\ 1, & 2 & & , N \end{array}$$

$\langle G \rangle$ with G <u>standard</u> graph

meaning $V = \{1, 2, \dots, N\}$:

$\langle G \rangle =$ (# vert in base 10) (give an edge)

(give another edge)

$\vdots$

e.g.

$$G = \begin{cases} V = \{1, \ldots, 1000000\} \\ E = \{\{2, 700318\}, \{4, 6\}\} \end{cases}$$

$$\langle G \rangle = 1000000\#2\#700318\#4\#6$$

Caution: If

$$n = \text{length } \langle G \rangle$$

$|V|$ could be 10^n

e.g. $V = \{1, \ldots, 1000\}$, $E = \emptyset$

$$\langle G \rangle = 1000$$

We could write

$$\langle G \rangle = \underbrace{(\text{\# of vertices})}_{N} \underbrace{101001c\ldots1}_{N\times N \text{ adjacency matrix}}$$

$$\langle G_1 \rangle = \underbrace{1000000}_{\text{optional}} \# 0000\ 1\quad 1\ 1\ 1\ldots$$

adjacency matrix

0

optional

Option:

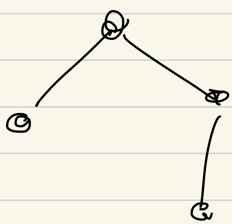
vertices is 17, could write

17 in base 1 : $\underbrace{11111111111111111}_{17\ 1\text{'s}}$

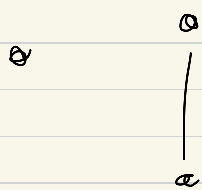
Unary ... to follow in ≈ week

CONNECTED-GRAPHS =

$\{ \langle G \rangle \mid$ G is a (standard) graph s.t. for any 2 vertices of G, there is a path between them $\}$



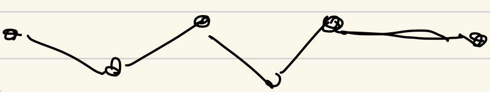
connected



not connected

path from V_1 to V_2

V_1



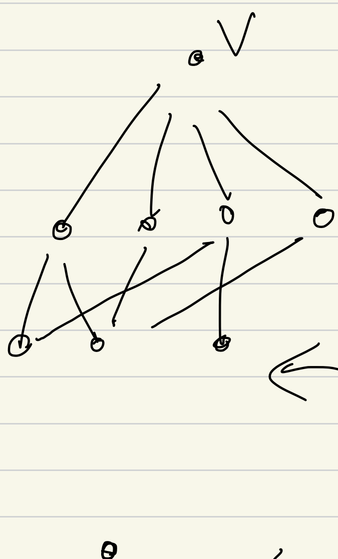
V_2

How we claim:

CONNECTED-GRAPHS $\in P$

Algorithm: pick any vertex

$v \in V = \{1, \ldots, N\}$;



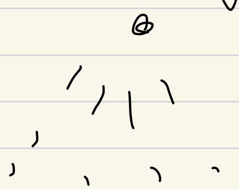
← which vertices are connected to v

← extra vertex

Say that $N = 10^6$

$E = \emptyset$

$v = 1$



is $v' = 2$ connected to v

$= 3$ " " "

⋮

$= 10^6$ " " v

?

Algorithm, to be poly-time, has to search through the edges

not " " vertices

NP = non-deterministic poly-time:

=

Turing machine: $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$

s.t. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

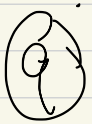
k-tape TM: same, but

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$$

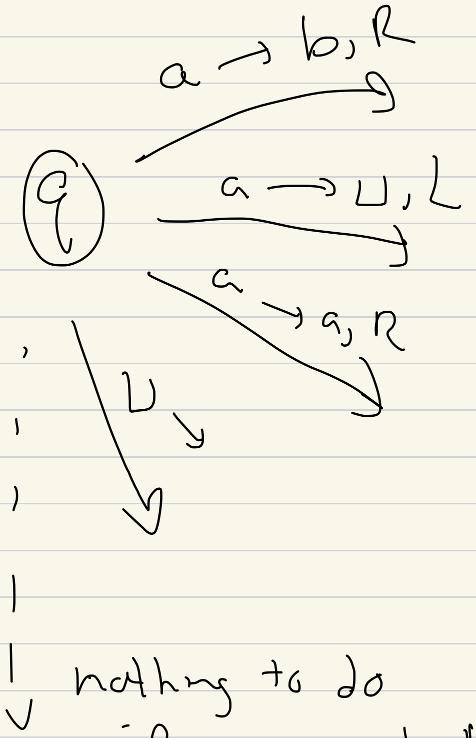
non-deterministic TM: (1-tape)

$$\delta : Q \times \Gamma \to \text{Power}\left(Q \times \Gamma \times \{L, R\}\right)$$

$$q \xrightarrow{a \to b, L} q' \quad \rightsquigarrow$$



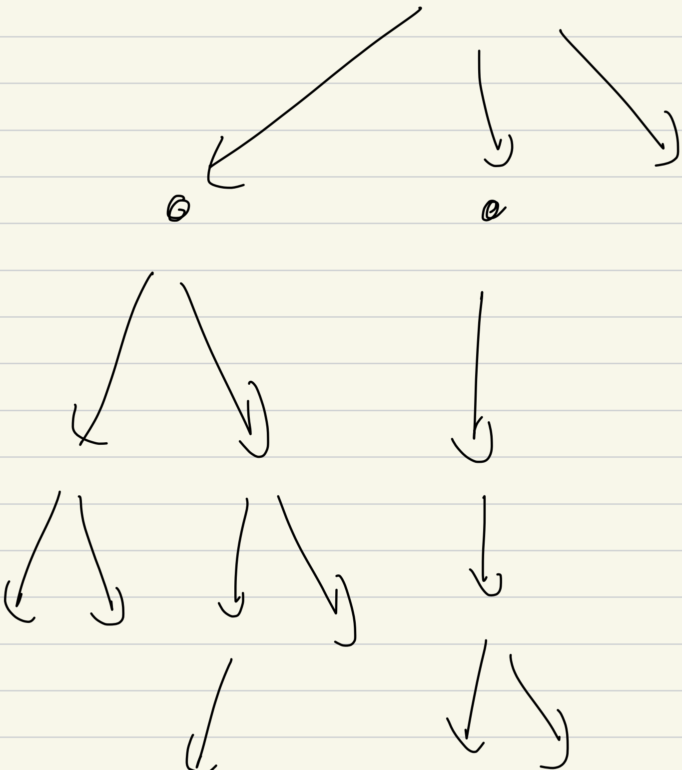
could be
nothing...



nothing to do
if you read ⊔

OR

initial config: state q_0 | a | b | b | a | ⊔ | ⊔ ...



step 2

accepting somewhere

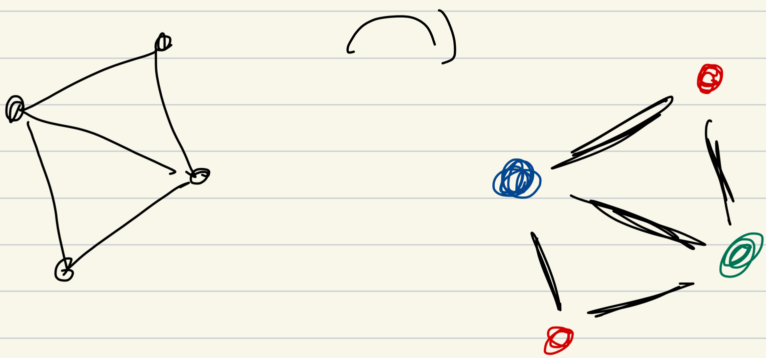
⋮

then the non-det computation accepts the input

Time: A non-deterministic TM, M, takes time $\leq f(n)$, where $f: \mathbb{N} \to \mathbb{N}$, if all possible computation paths, on any input of length n, halt or stop in $\leq f(n)$ steps.

$$3\text{-COLOR} = \{ \langle G \rangle \mid G \text{ is a graph that is 3-colourable} \}$$

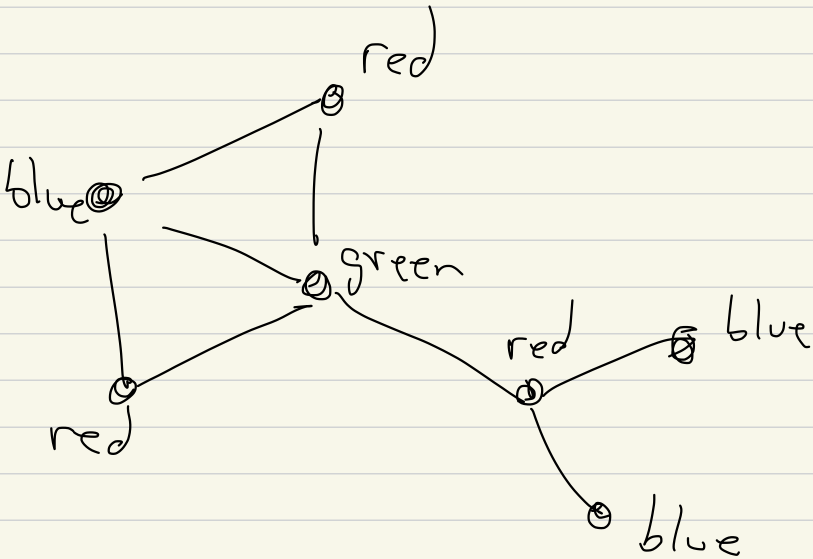
3-colourable: colour the vertices with 3-colours

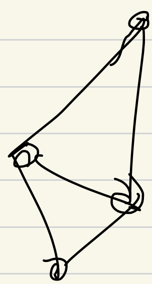


Formally $V \longrightarrow \{\text{red}, \text{green}, \text{blue}\}$

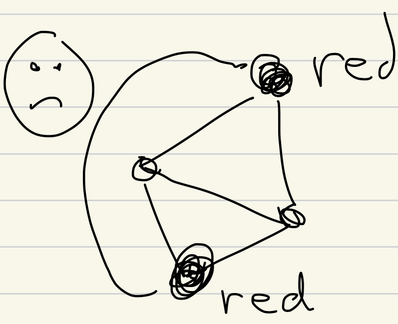
s.t. each edge has different colours at its endpoints.

E.g.





3-colorable

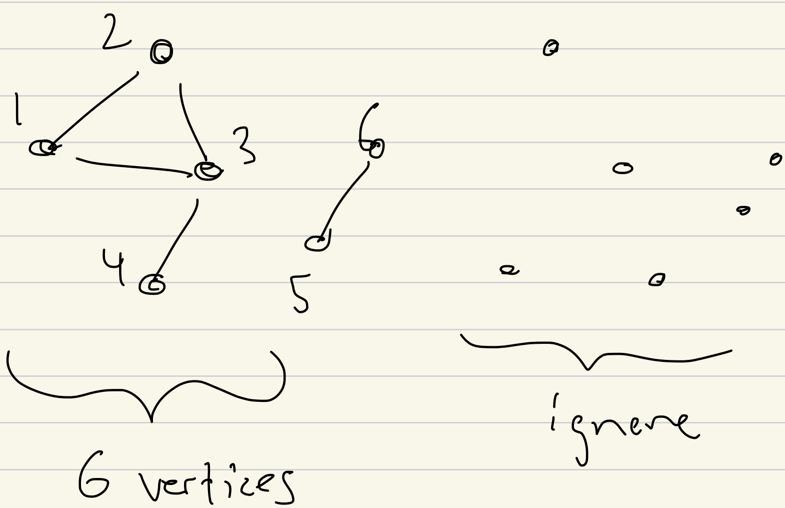


3-COLOR can be decided

by a poly-time

non-deterministic algorithm:

Start



write down a sequence

Red	Red	Blue	Green	Green	Blue
1	2	3	4	5	6