Computation

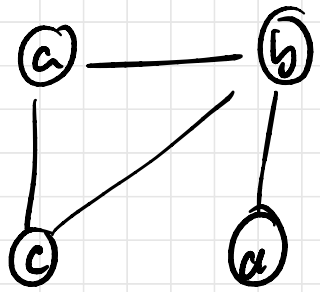
CSCE 310ff – Fall 2021

`.*` `[a-z]*`

regular expressions

`[0-9][0-9][0-9]`

`4[0-9][0-9]*`



$n = 4$

$a \rightarrow 00$

$b \rightarrow 01$

$c \rightarrow 10$

$d \rightarrow 11$

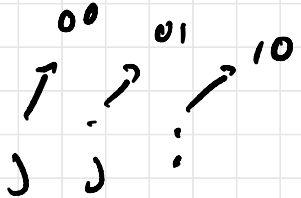
edge

00 : 01

00 : 10

01 : 10

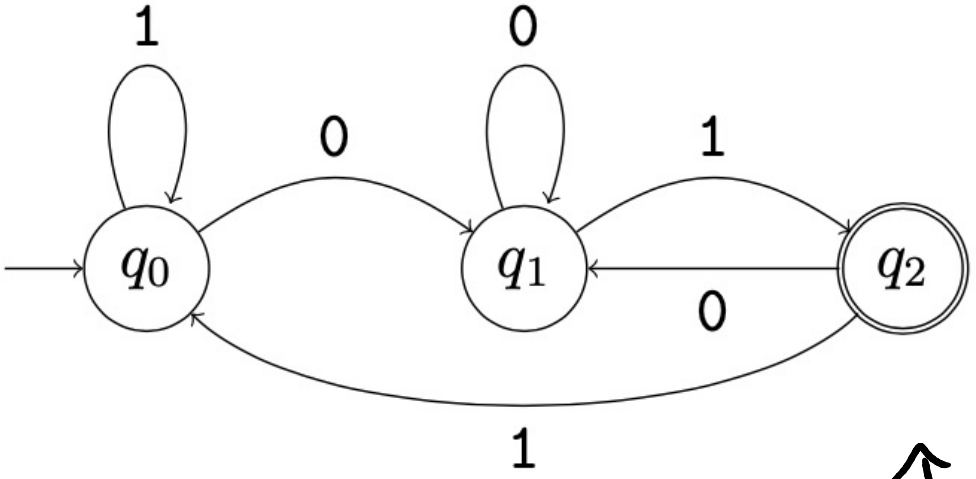
01 : 11



$\langle G7 \rangle = \langle n \rangle$; 00:01, 00:10, 01:10, 01:11

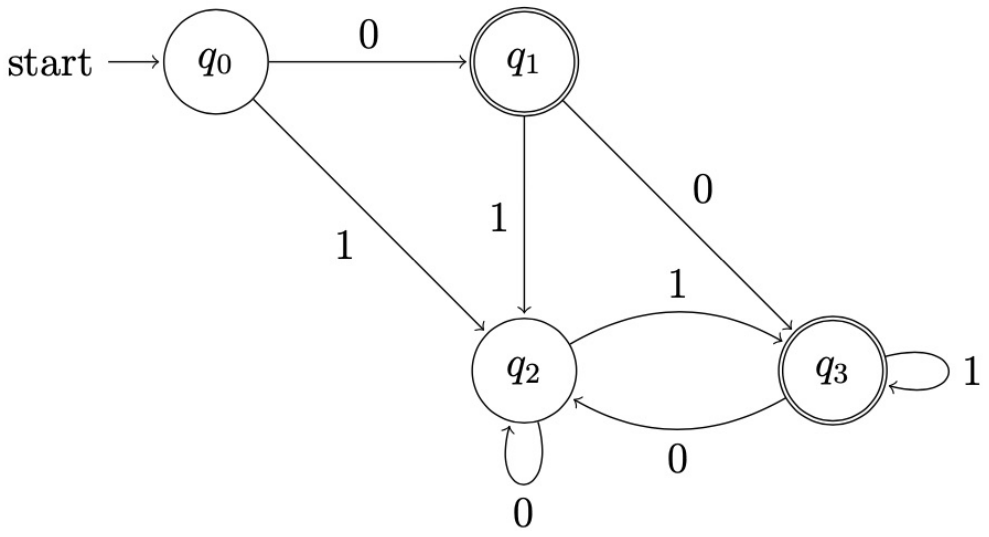
$= 100$; 00:01, 00:10, 01:10, 01:11

Input	result
001010	reject $\notin L$
0001	accept $\in L$
10101	accept $\in L$
1111	reject $\notin L$
111101	accept
01	accept
1	reject
λ	reject



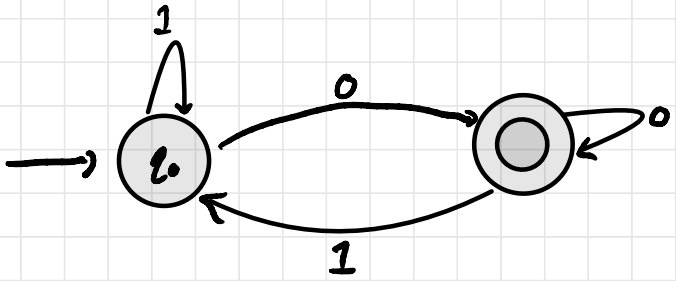
$\equiv$

$\Sigma^* 01$ regular expression

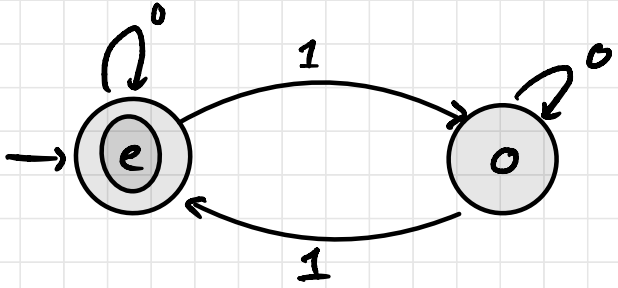


in L	$\notin L$
0	π
11	1
00	01
001	

$L_{even} = \{ x \mid x$ is a binary representation of an even number$\}$



$L_{parity} = \{ x \mid$ parity of x is even$\}$



$$L = \{ 0^n 1^n \mid n \in \mathbb{Z} \}$$

n 0s/n 1s

λ

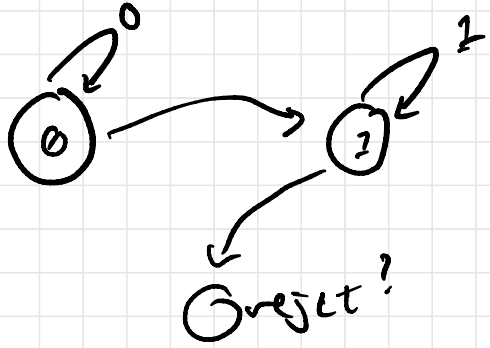
01

0011

000111

$0101 \notin L$

$1100 \notin L$



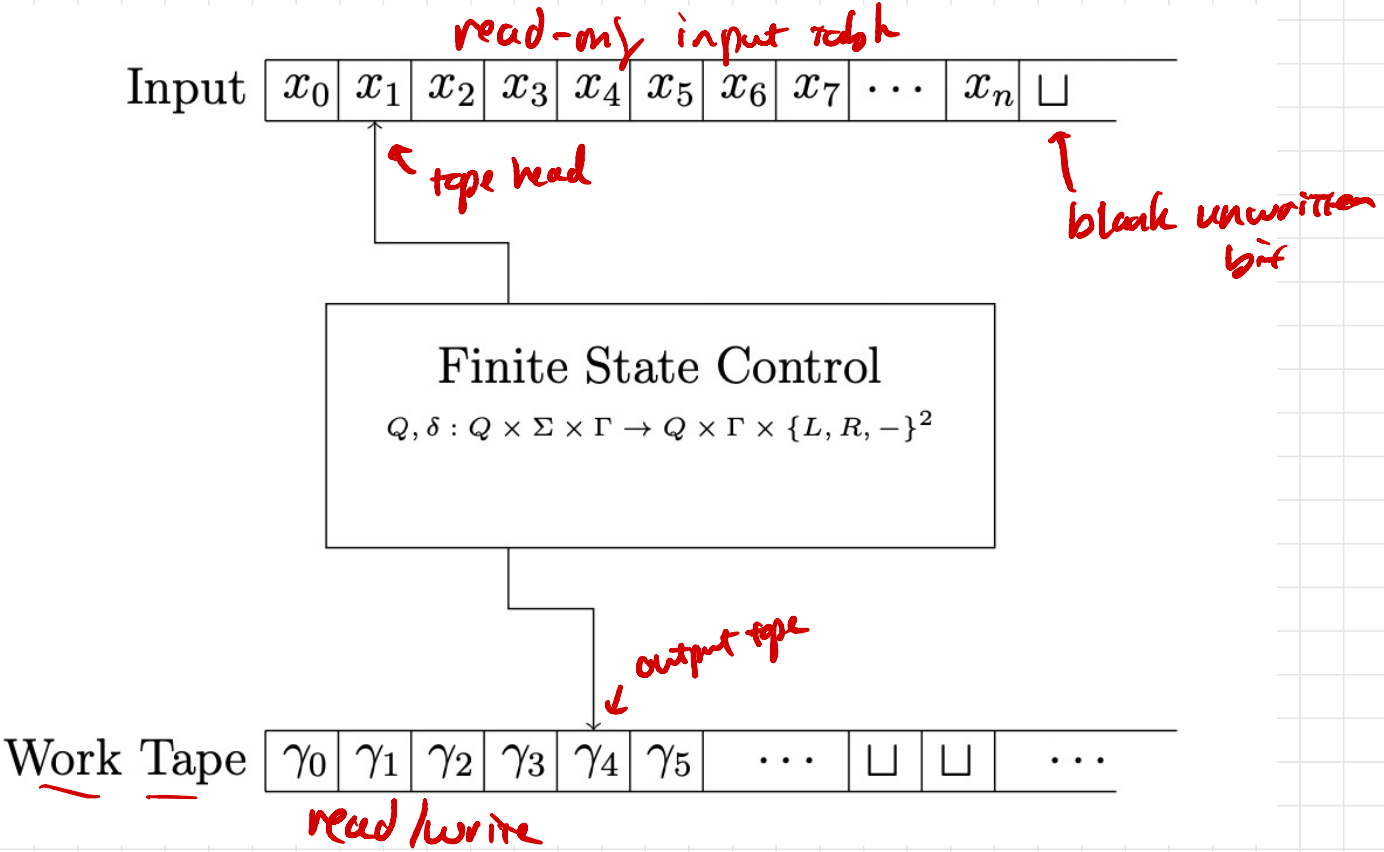
$$L = \{ 0^n 1^n \mid n \in \mathbb{Z} \}$$

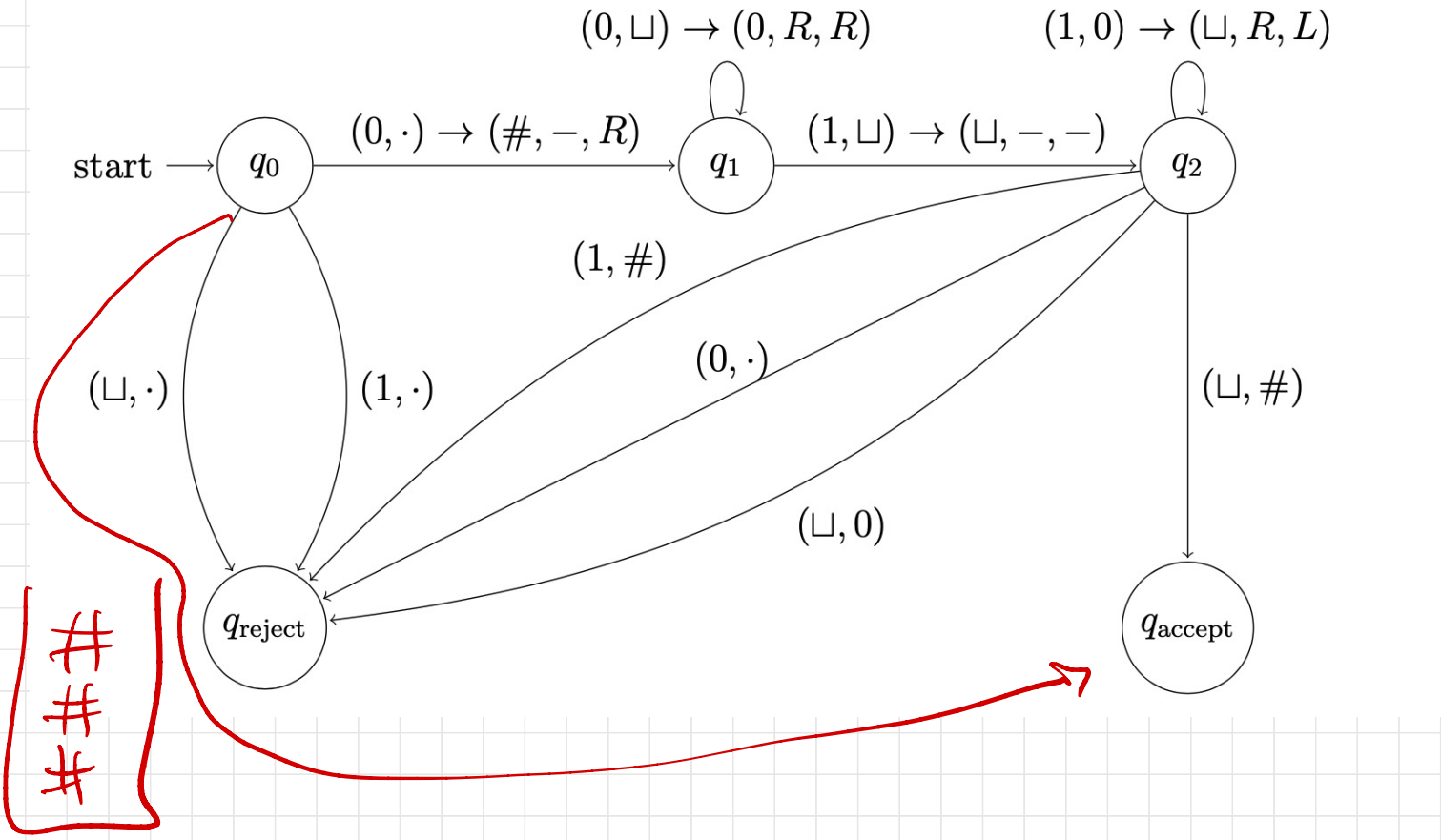
Idea: push 0's until you see a 1

↓

mode 2: pop 0s, match each 1

FSA → + stack → + Queue


read-only input tape
Input
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 $\cdots$ x_n $\sqcup$
tape head
blank unwritten bit
Finite State Control
$Q, \delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma \times \{L, R, -\}^2$
output tape
Work Tape
γ_0 γ_1 γ_2 γ_3 γ_4 γ_5 $\cdots$ $\sqcup$ $\sqcup$ $\cdots$
read/write


$(0, \sqcup) \to (0, R, R)$
$(1, 0) \to (\sqcup, R, L)$
start
q_0
$(0, \cdot) \to (\#, -, R)$
q_1
$(1, \sqcup) \to (\sqcup, -, -)$
q_2
$(1, \#)$
$(\sqcup, \cdot)$
$(1, \cdot)$
$(0, \cdot)$
$(\sqcup, \#)$
$(\sqcup, 0)$
q_{reject}
q_{accept}
#
#
#

Claim: Turing Machines are countable

- Turing Machines have a finite binary representation

$$\langle M \rangle$$

- you can enumerate (write down) list of all possible TMs in a sequence

$$M_1 \; M_2 \; M_3 \; \ldots$$

$\lambda, \cancel{0}, 1, 00, \cancel{01}, 10, 11, 000 \ldots$

$\downarrow$

$M_1 \; M_2 \; M_3 \ldots$

$\therefore$ TMs are infinitely countable!

- We will show Languages are infinitely uncountable!

By way of contradiction suppose languages are countable

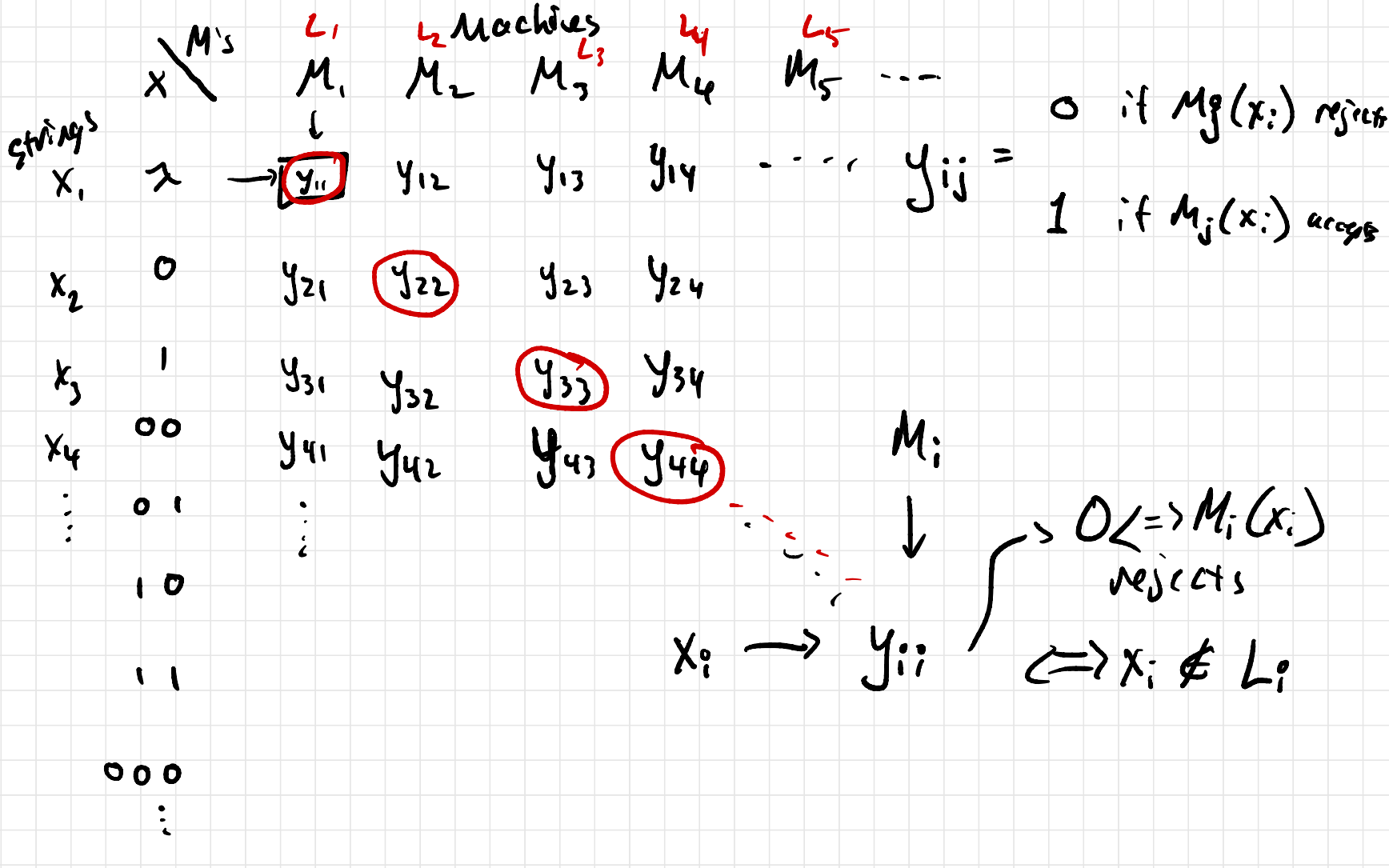
$L_1\ L_2\ L_3\ \cdots$

Suppose we align Th 2 enumerations:

$M_1\ \ M_2\ \ M_3\ \cdots$

Decides $\downarrow$

$L_1\ \ L_2\ \ L_3\ \cdots$


M's
Machines
L_1
L_2
L_3
L_4
L_5
x
M_1 M_2 M_3 M_4 M_5 ---
strings
x_1 λ y_{11} y_{12} y_{13} y_{14} ---
x_2 0 y_{21} y_{22} y_{23} y_{24}
x_3 1 y_{31} y_{32} y_{33} y_{34}
x_4 00 y_{41} y_{42} y_{43} y_{44}
01
10
11
000
y_{ij} =
0 if $M_j(x_i)$ rejects
1 if $M_j(x_i)$ accepts
M_i
$x_i \rightarrow y_{ii}$
$0 \Leftrightarrow M_i(x_i)$ rejects
$\Leftrightarrow x_i \notin L_i$

$$L = \{ x_i \mid x_i \text{ is a string and } y_{ii} = 0 \}$$

$x_i \in L \iff x_i \notin L_i$ for each L_i

$\iff L \neq L_i$ for any i in our supposed enumeration

$\iff L$ is not in our enumeration

$\iff$ no such enum exists

Given: A Turing machine M, does it halt on all inputs or

Does exist an input x $M(x)$ end up in an int loop.

No such TM decides this question exists!

Pf BWC assume the "Halting Problem" is decidable

$\Rightarrow \exists$ TM $H(\langle M, x \rangle) =$ accepts if $M(x)$ halts

rejects if $M(x) \to \infty$

- We will run a machine on itself...

$$M(M)$$

- Create a new TM: Q that takes a machine M as input

```
Q(M)
  Input: A TM M
   if H(<M, M>) rejects:
    L stop (halt) accept
   else if H(<M M>) accepts:
    L while(1): noop
```

$Q(\langle Q \rangle)$

Input: Q

if $H(\langle Q, Q \rangle)$ reject:

L stop, accept

else:

while(1): loop

$Q(\langle Q \rangle)$ halts $\overset{?}{\iff}$ $H(\langle Q, Q \rangle)$ rejects $\iff$ $Q(Q)$ does not halt.

contradiction.

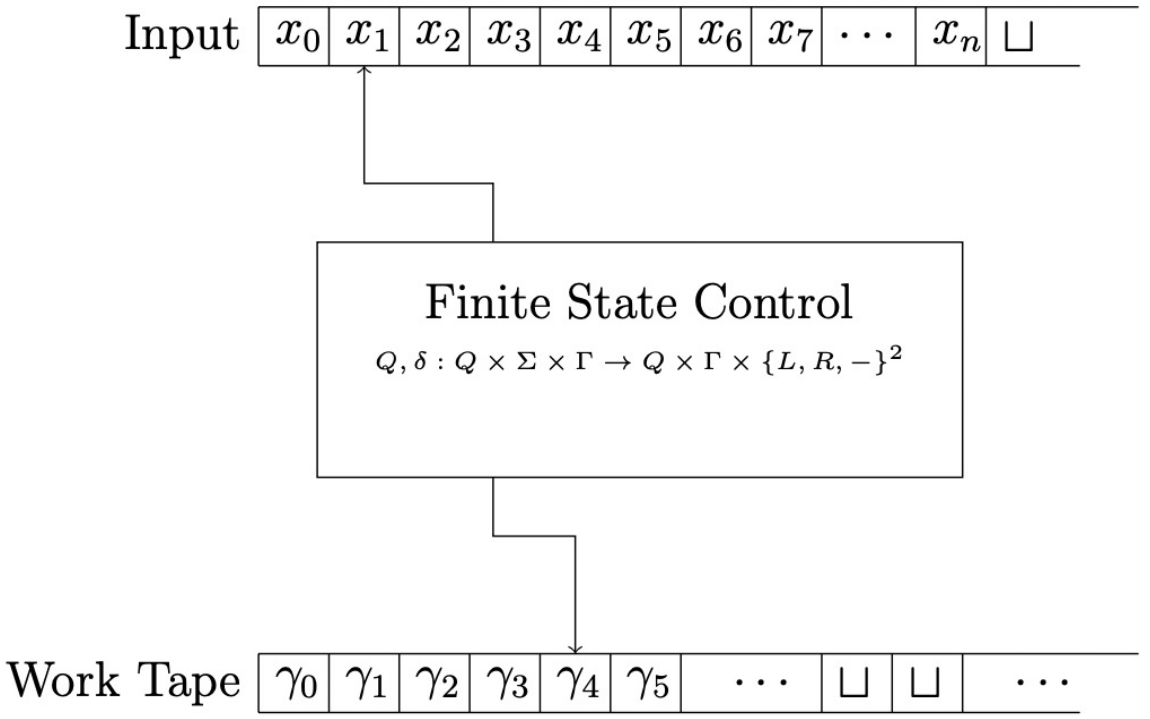
$\sqrt{2}$ is irrational

$\sqrt{2} = \frac{a}{b}$ lowest terms

$2 = \frac{a^2}{b^2}$ ...

$2 \mid a^2 \Rightarrow 2 \mid a$

$2b^2 = a^2 = 2k(?)$


Input
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 $\cdots$ x_n $\sqcup$
Finite State Control
$Q, \delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma \times \{L, R, -\}^2$
Work Tape
γ_0 γ_1 γ_2 γ_3 γ_4 γ_5 $\cdots$ $\sqcup$ $\sqcup$ $\cdots$

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

Satisfiable if there exists an assignment $x_1 \cdots x_n$ such that it evaluates to true

x_1	x_2	x_3	value
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Given n variables

There are 2^n possible assignments

$O(2^n)$ = exponential

Nondeterministic SAT

Input: a propositional formula $\phi(\vec{x})$ on variables $x_1 \cdots x_n$

Output: accept if ϕ is satisfiable, reject if NOT

1) Guess a truth assignment $\vec{x} = x_1 \cdots x_n$ $O(n)$ non-det.

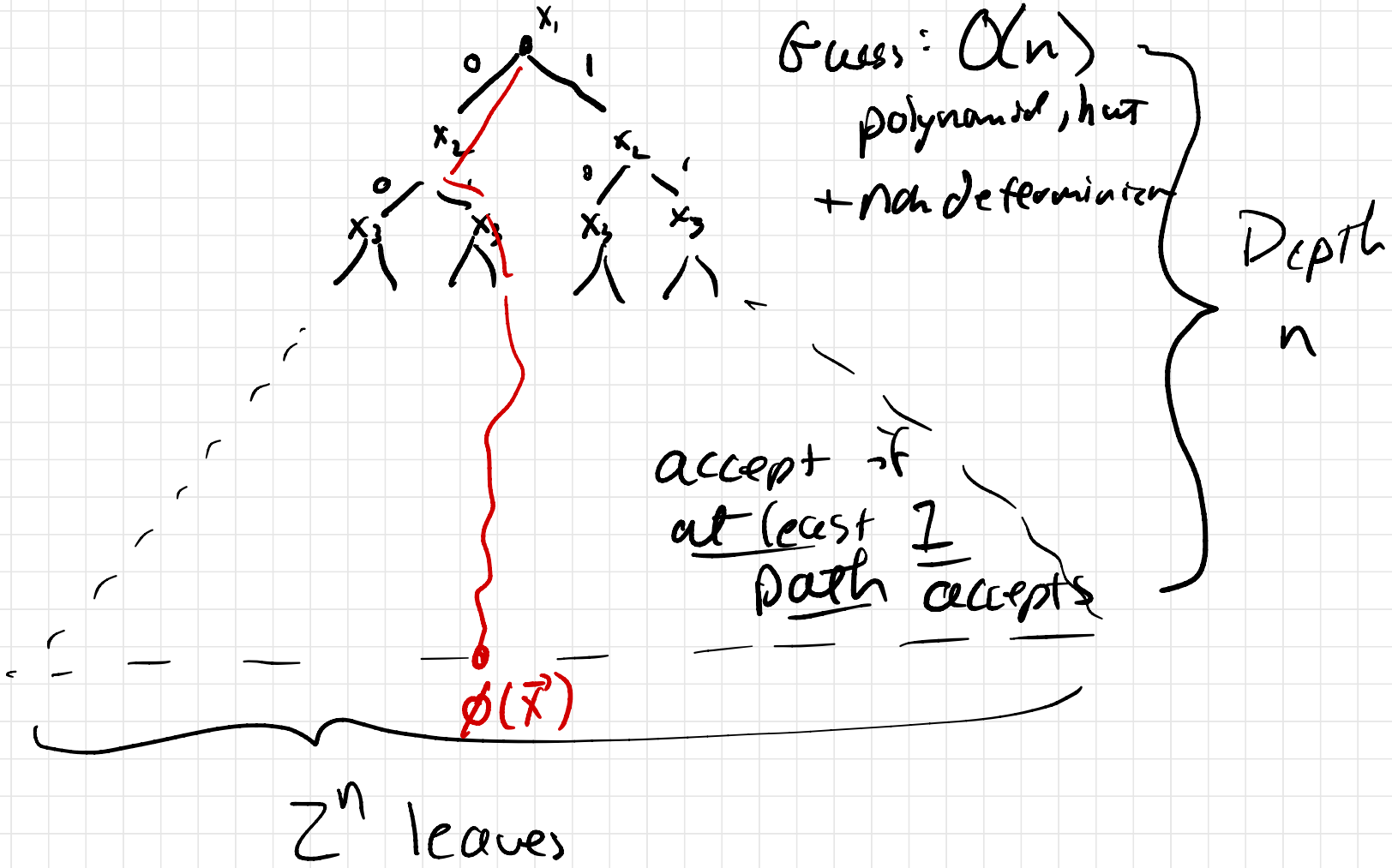
2) Verify: if $\phi(\vec{x}) \equiv 1$ → $O(n^2)$ deterministic

accept

else

reject

NP


Guess: O(n)
polynomial, but
+ nondeterminism
Depth
n
accept if
at least 1
path accepts
$\phi(\vec{x})$
2^n leaves

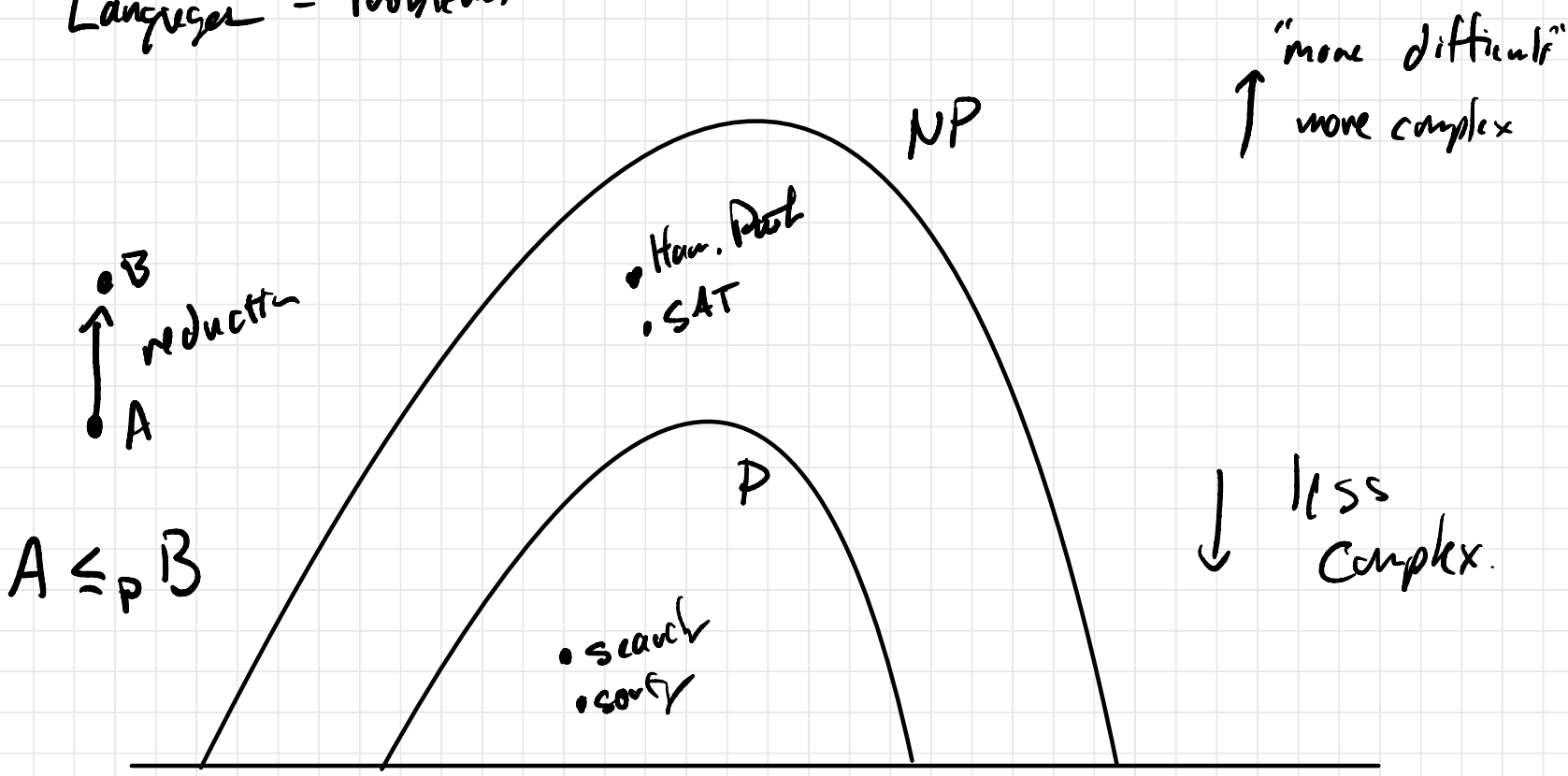
Deterministic / Brute force solution: $O(2^n)$

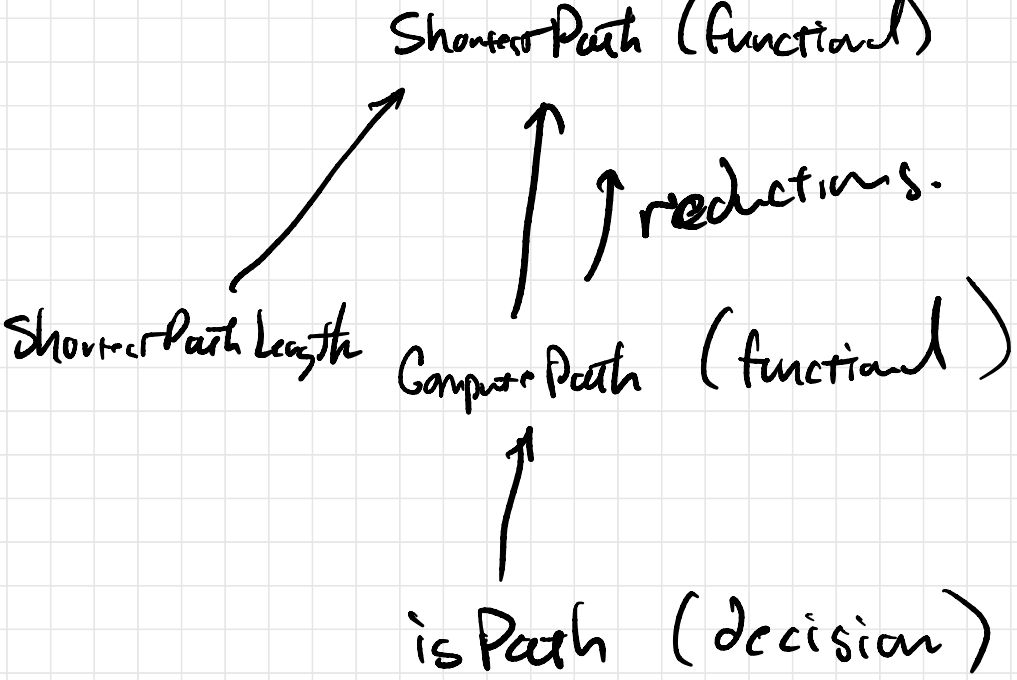
Non-deterministic: $O(n)$

$$\exists\, x_1\, x_2 \cdots x_n \left[\phi(x_1 \cdots x_n) \equiv 1\right]$$

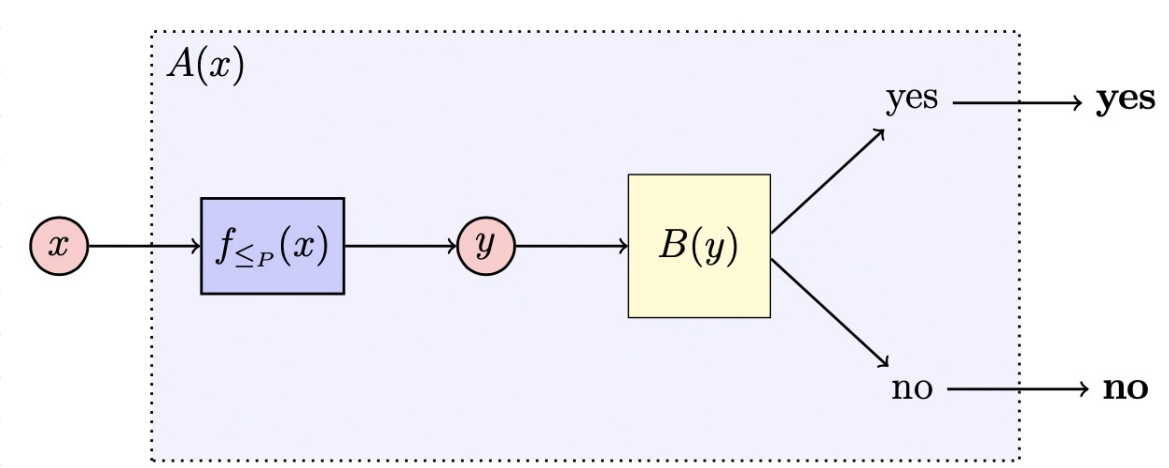
$\exists$ — there exists

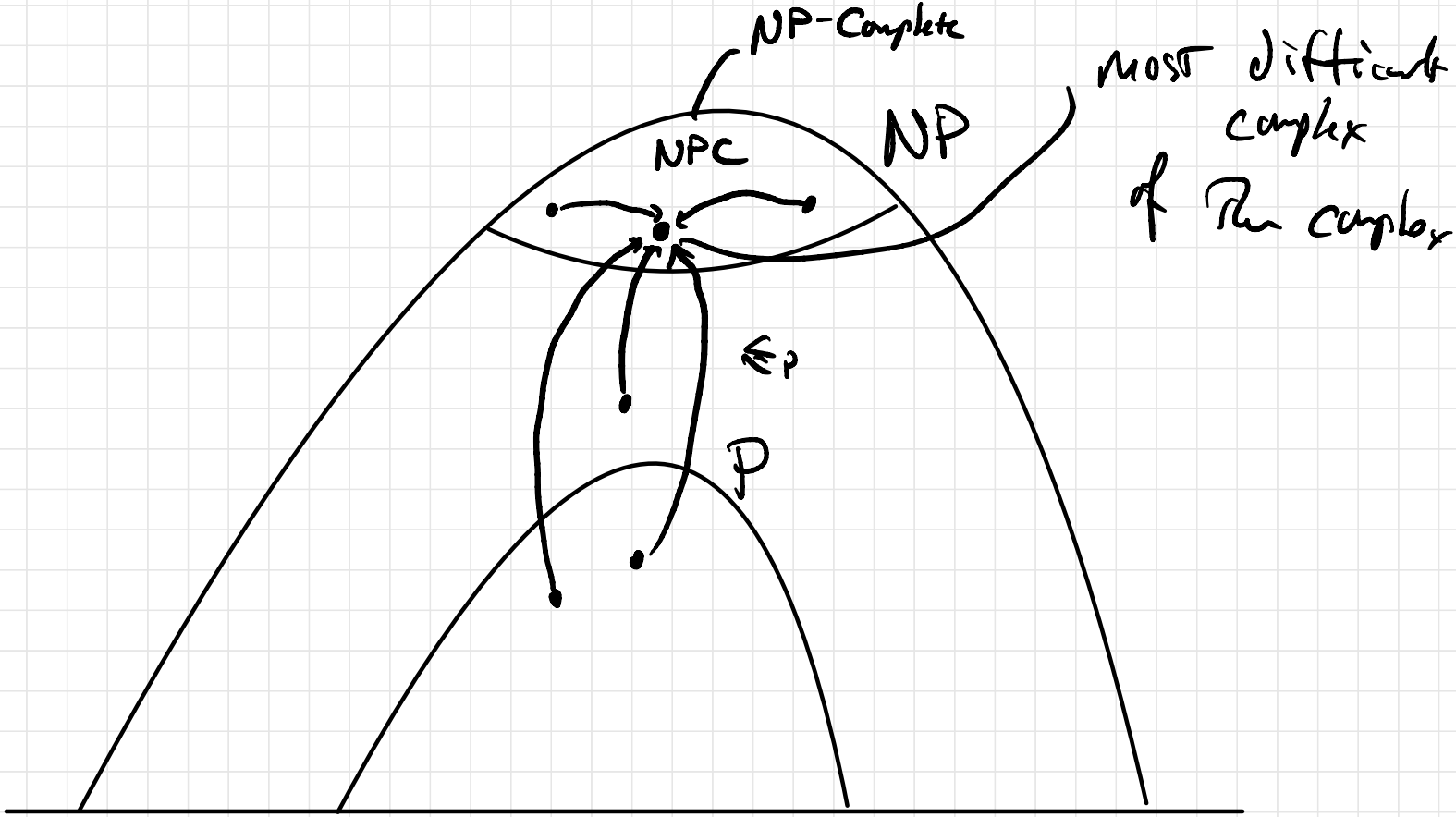
Languages = Problems

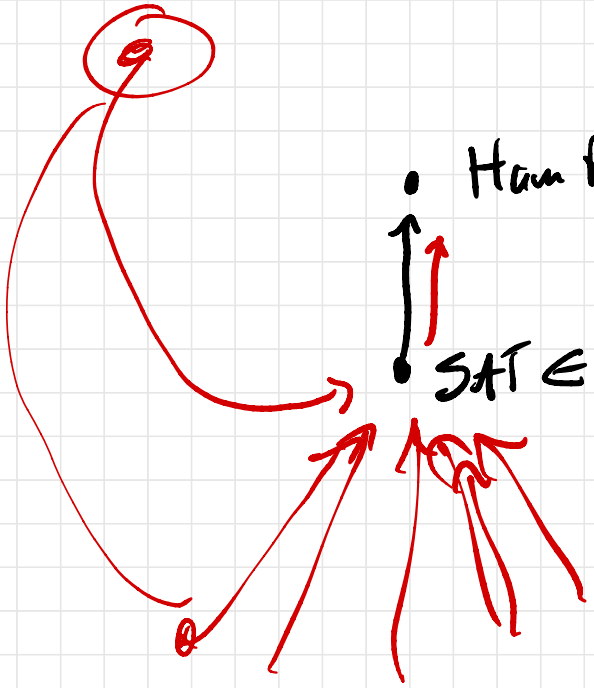



ShortestPath (functional)
reductions.
ShortestPathLength
ComputePath (functional)
isPath (decision)

- Solutions to B can be used or adapted to solve A
- B is at least as complex / difficult as A
- A is no more difficult or complex than B
- $A \leq_p B$

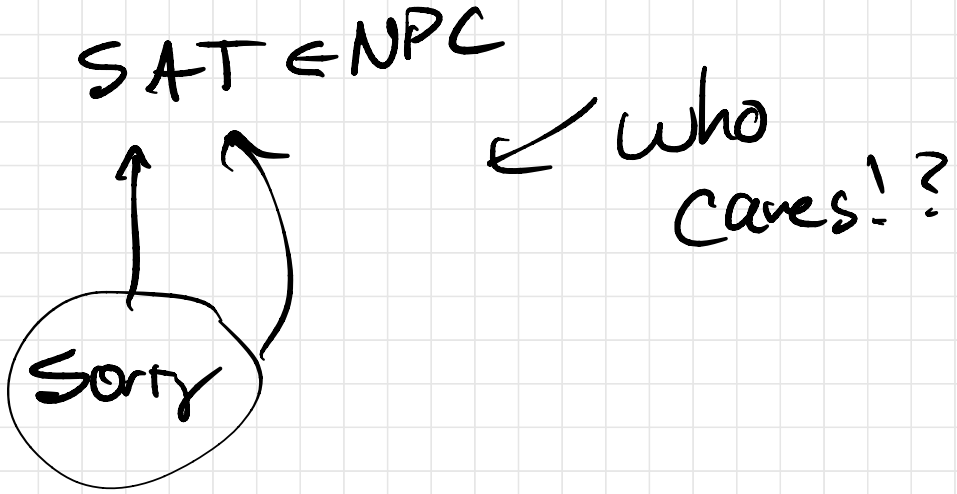

$A(x)$
x
$f_{\leq P}(x)$
y
$B(y)$
yes
yes
no
no

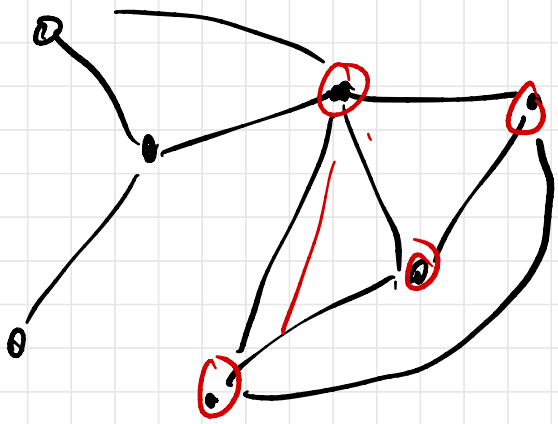

NP-Complete
NP
NPC
$\leq_P$
P
most difficult
complex
of the complex



Ham Path $\in$ NPC

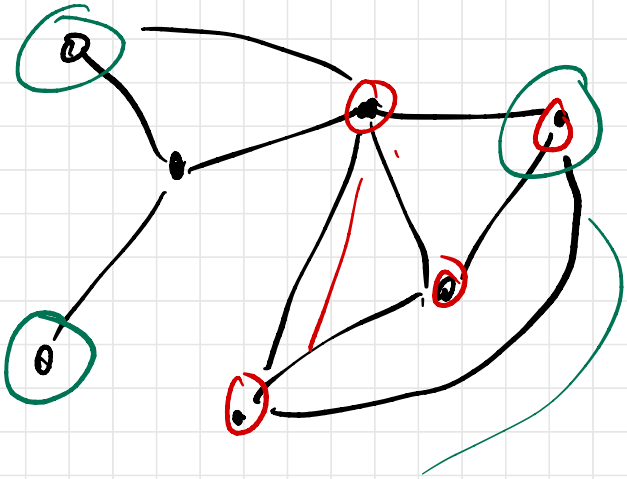
SAT $\in$ NPC $\Rightarrow$ Ham Path $\in$ NPC


SAT ∈ NPC
Who cares!?
Sorry



Clique: a subset of vertices that are all connected

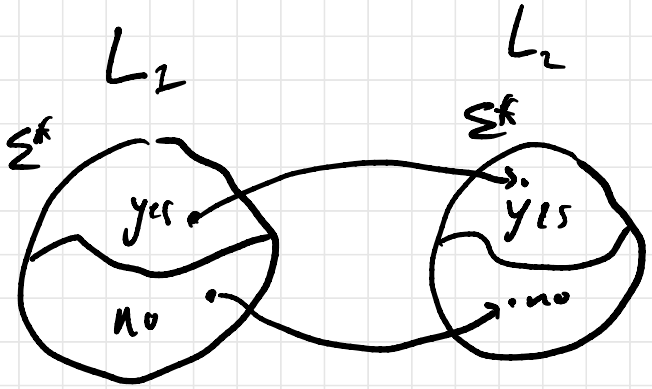
$K = 4$



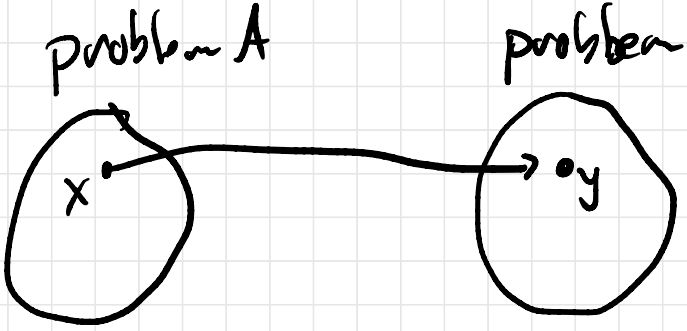
Independent Set:

a subset of vertices s.t.

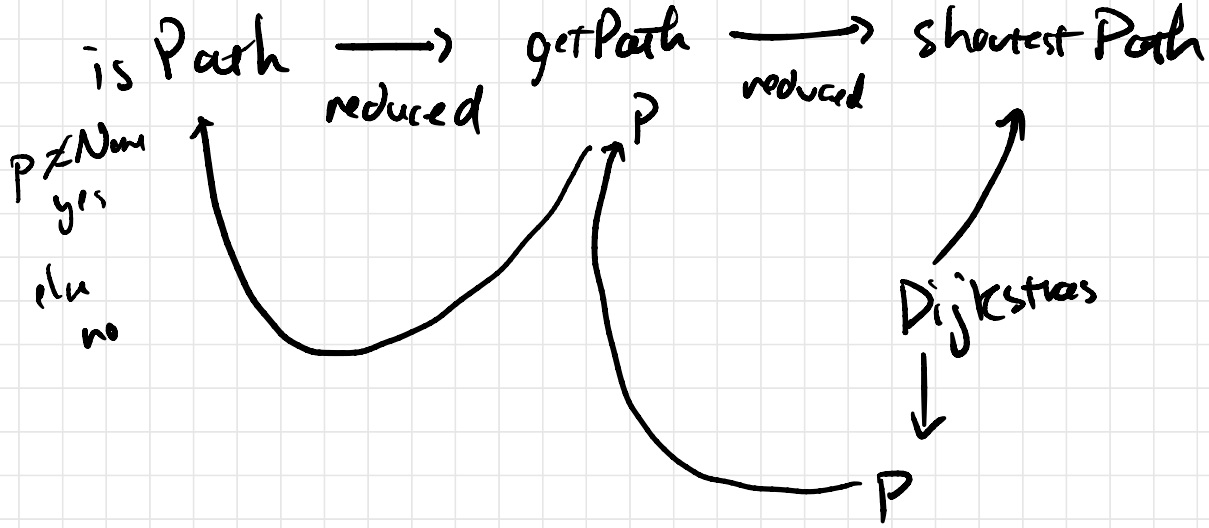
~~that~~ none are connected



String x → string y



↳ $B(x) = \text{yes} \iff A(x) = \text{yes}$


isPath
reduced
getPath
reduced
shortestPath
P
P
Dijkstras
P≠Num
yes
else
no

"harder"

shortestPath

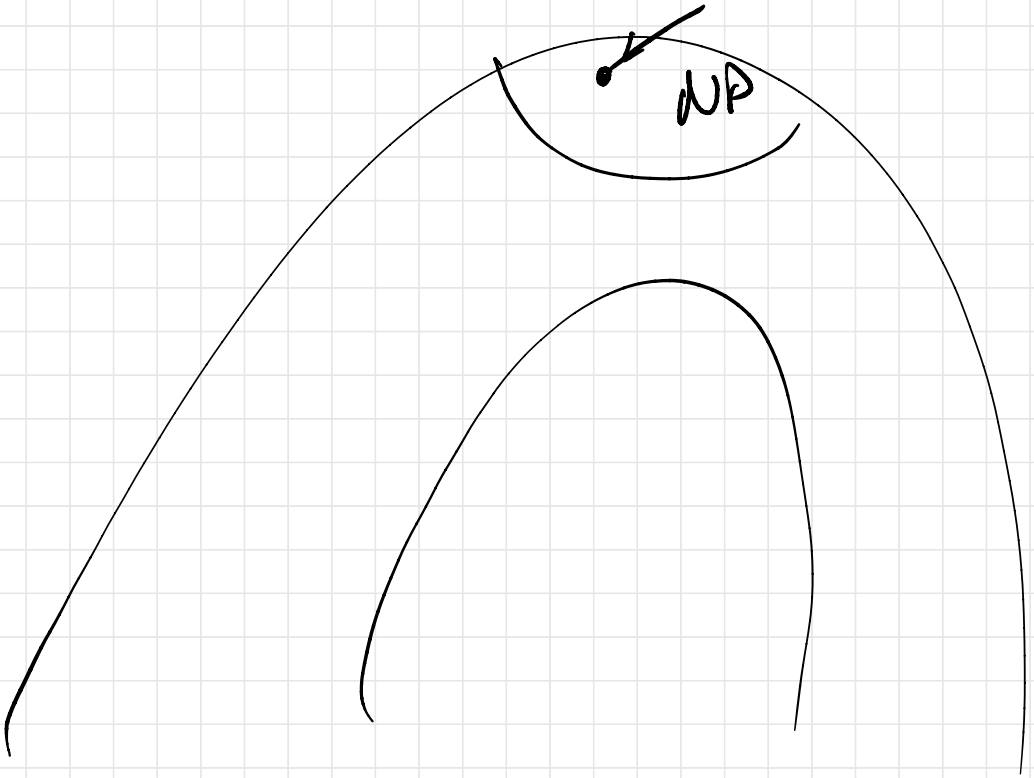
$\nearrow \leq_p$

longPath

$\nearrow \leq_p$

isPath

"easier"

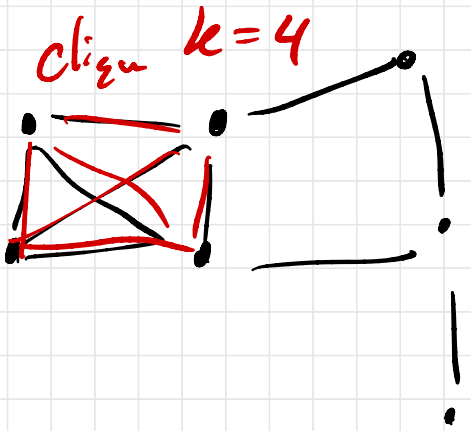

NP

A clique in a graph $G = (V, E)$ is a subset of vertices $C \subseteq V$ that are all connected

$$\forall x, y \in C \left[(x, y) \in E \right]$$

Finding the maximal clique is an NP-Complete problem.

max possible: $|V| = n$ if $G = K_n$


clique
k=4

$Clique = \{\langle G, k\rangle \mid$ There is a clique of size k in $G\}$

(at least)

$C(G, k)$

for $k = n \dots 1$

  if $C(G, k)$ accepts

    output k

① Input: A graph G, k

Output: accept if There is a clique of size k in G

① Guess a subset $C \subseteq V$ of size k $(|C| = k)$

② Verify:

for each pair $x, y \in C$

  if $(x, y) \notin E$

    reject

accept

a) Clearly if is non-det.

b) non-det: $O(k) = O(n)$

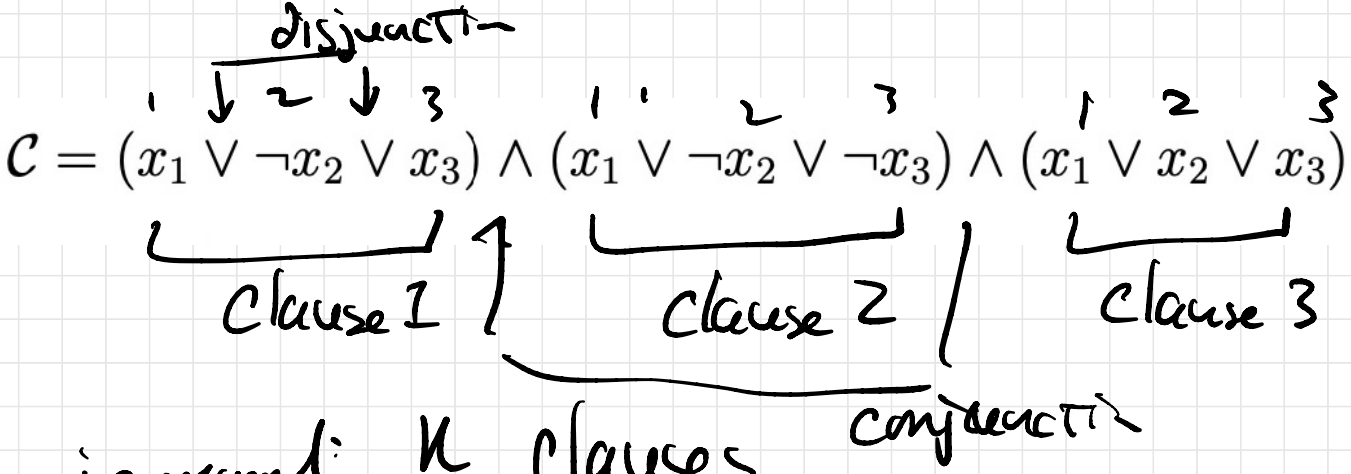
verify: $O(k^2) = O(n^2)$

$k \in O(n)$

(2) SAT:

3-CNF-SAT

- 3 → each "clause" has at most 3 variables or their negations
- CNF → Conjunctive normal form



in general: k clauses

each clause only has ≤ 3 variables

③ NPC → NPC such that

ϕ → G

3CNF formula → graph

ϕ is satisfiable iff G has a clique

k = # of clauses = size of the clique

$Algo(G)$ = yes if it has a clique of size k

SAT Solver

① SAT → 3-CNF, ϕ

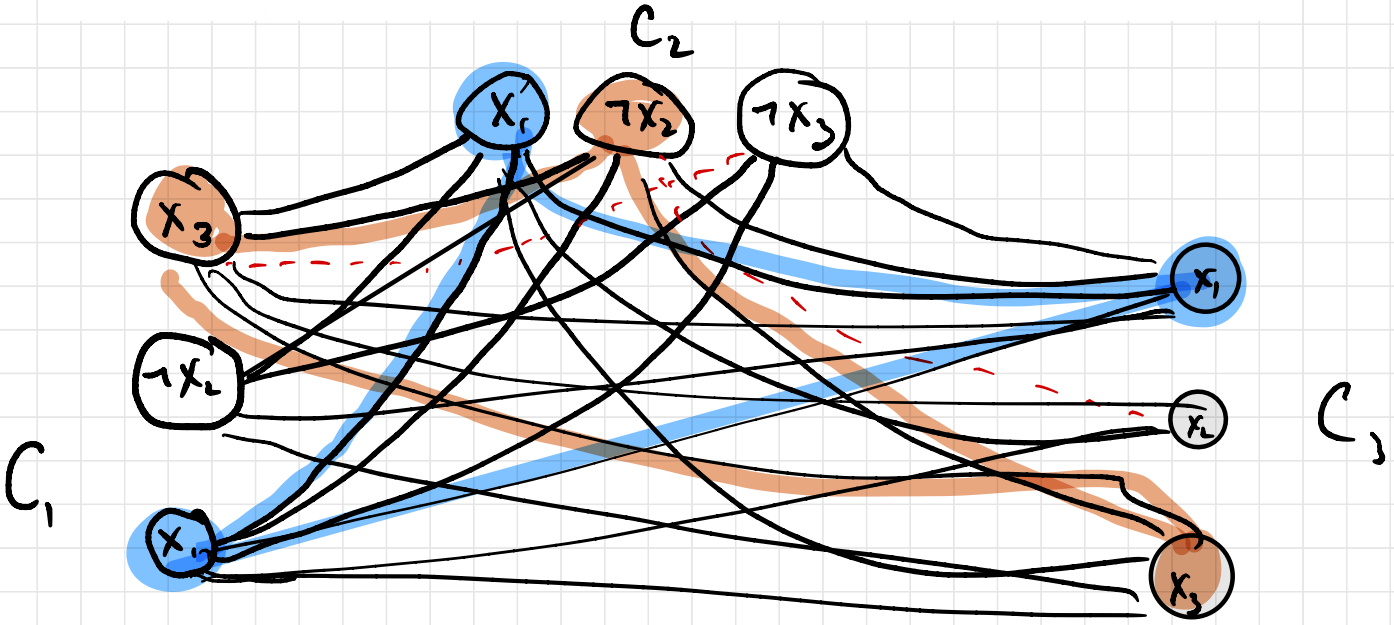
② Map $\phi \rightarrow G$

③ if $Algo(G)$ accepts:
accept else reject

④ Given $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$

- for each clause $C_i = (x_1^i \vee x_2^i \vee x_3^i)$
  create 3 vertices, v_1^i v_2^i v_3^i (vertices)
- edges: Draw edge if both:
  a). if x's are in different clauses
  b). they are not negations of each other.

$$C = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3)$$



④ ϕ has a satisfy assignt iff G has a clique of size k

⑤ Claim: reduction is polytime

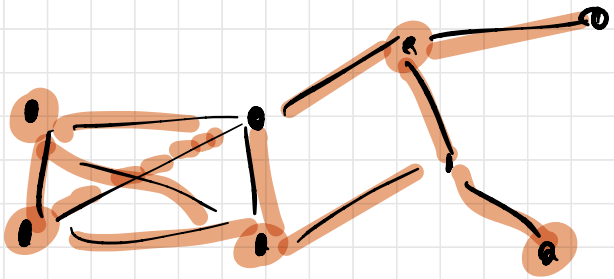
$$\phi = C_1 \wedge C_2 \cdots \wedge C_k$$

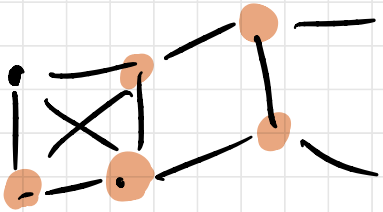
$$\downarrow$$

$$G = (V, E)$$

$$|V| = 3k \in O(n)$$

$$|E| \in O(n^2)$$



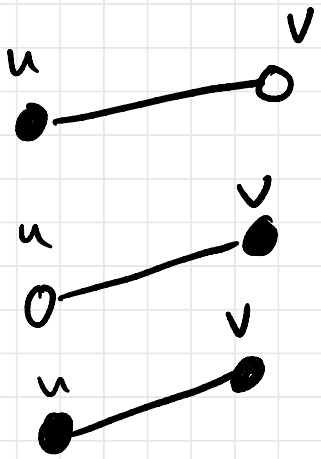


Vertex Cover

5 ?

4 ?

A vertex cover of a graph $G=(V,E)$... is a subset of vertices $V' \subseteq V$ that "covers" all edges



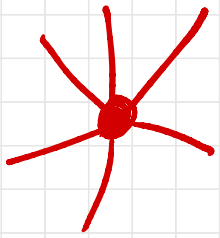
V' is a vertex cover iff

$$\forall e=(u,v) \in E \left[u \in V \vee v \in V \right]$$

Fact: The minimum vertex cover problem (find a V.C. of minimal cardinality) is NP-C

Decision version:

$\langle G, k \rangle$: G has a vertex cover of cardinality k



(0) $1 \leq k \leq n$

1 Vertex Cover is in NP:

Input: A graph $G = (V, E)$, $k \in \mathbb{Z}$, $1 \leq k \leq n$

Output: Accept iff G has a V. cover of size k

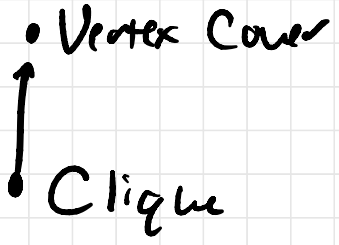
① Guess a subset $V' \subseteq V$ of size k — non-det. $O(k) = O(n)$

② Verify: — deterministic $O(n^2)$

```
foreach e = (u,v) ∈ E
    if u ∉ V' ∧ v ∉ V'
        reject
accept
```

[2] Choose an NP-C problem to reduce from

Choose: Clique



NPC → NPC

Clique $\leq_p$ Vertex Cover

3 Create a map:

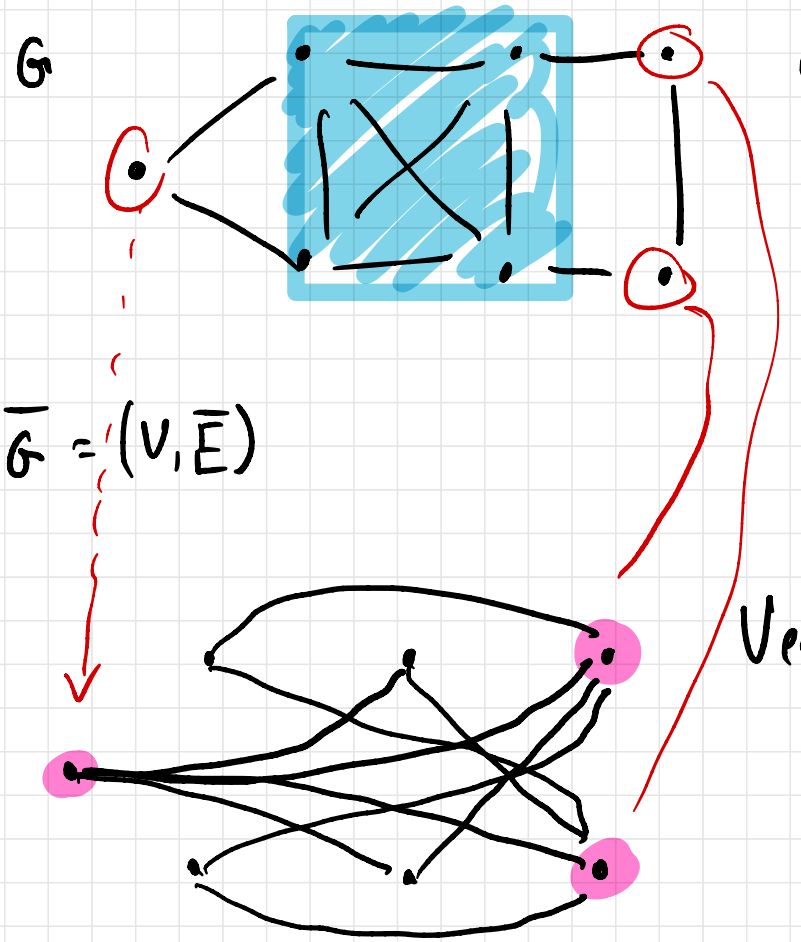
$$G \longrightarrow G'$$
$$?$$

G has a clique of size $k \iff G'$ has a vertex cover of size k'

Given: $\langle G, k \rangle \longrightarrow \langle \bar{G}, n-k \rangle$

Property: G has a clique of size k iff ($\Leftrightarrow$)

$\bar{G}$ has a vertex cover of size $n-k$ $(n = |V|)$



Clique of size $k = 4$

Vertex cover of size $n - k$ $7 - 4 = 3$

[4] Want to show:

G has a clique of size $k \iff \bar{G}$ has a V. cover of size $n-k$

($\Rightarrow$) Let C be a clique in G of size $|C| = k$

Consider an edge $e = (u, v)$ in $\bar{E}$:

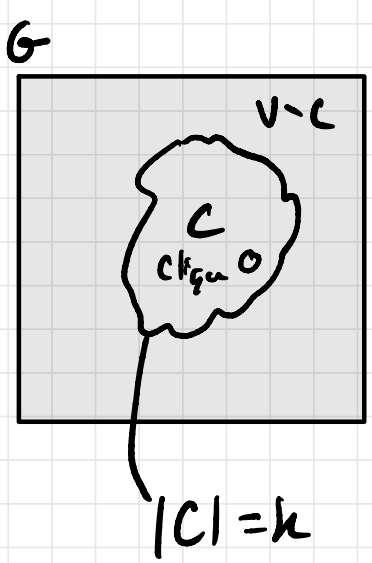
We need to show it is covered by some vertex

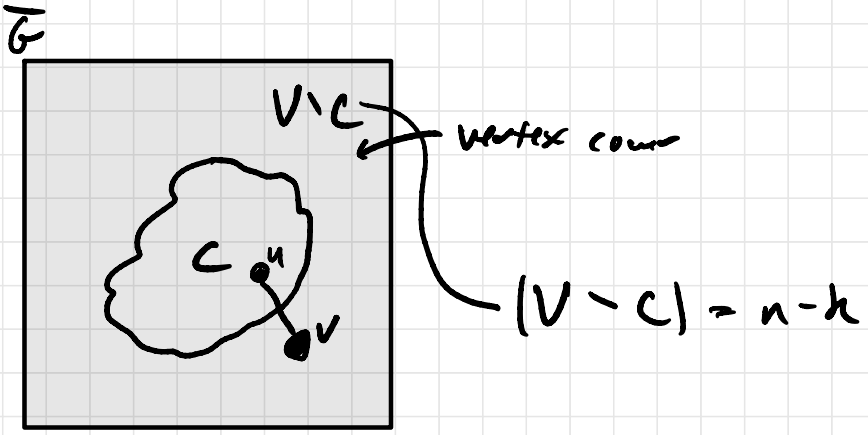
$e = (u,v) \in \bar{E} \Rightarrow (u,v) \notin E$ def. of complement graph

$\Rightarrow u \notin C \vee v \notin C$ (or both)

$\Rightarrow u \in V \setminus C$ or $v \in V \setminus C$

$\Rightarrow (u, v)$ in $\bar{G}$ is covered by $V \setminus C$


G
V\C
C
clique
|C| = k


G
V\C
vertex cover
C
u
v
|V\C| = n-k

G has a clique of size $k \iff \bar{G}$ has a V. cover of size $n-k$

($\Leftarrow$) Let $W \subseteq V$ be a vertex cover in $\bar{G}$ of size $n-k$

$P \rightarrow Q \equiv \neg Q \rightarrow \neg P$

$$\forall u,v \in V \left[(u,v) \in \bar{E} \Rightarrow u \in W \vee v \in W \right]$$

contrapositive:

$$W = V \setminus C$$

$$\forall u,v \in V \left[u \notin W \wedge v \notin W \Rightarrow (u,v) \in E \right]$$

$\Rightarrow$ everything outside of $W = V \setminus C$ is connected.

$V \setminus C$ is a clique

$\Rightarrow$ "

side of empty outside $V-C_1$

$|V|$ $\quad$ $|V - C|$

$\|$ $\quad$ $\|$

n $\quad$ $n-k$

$n - (n-k)$

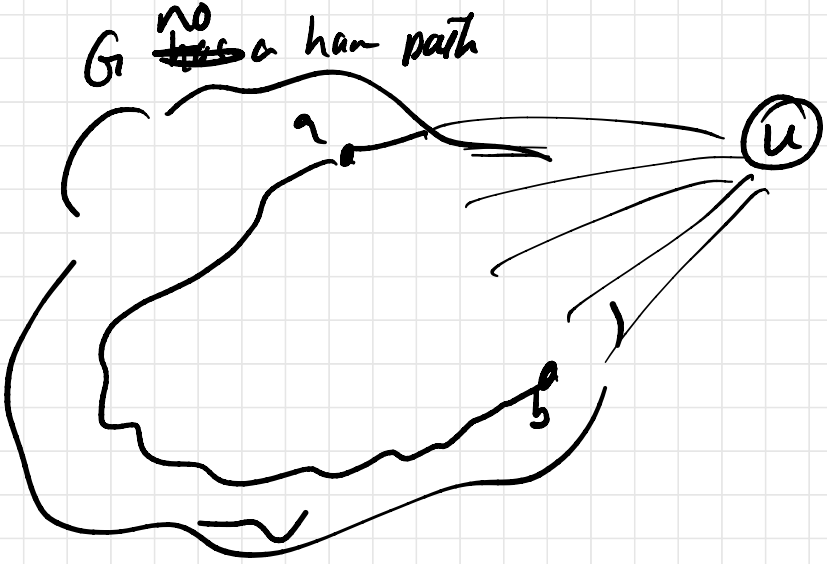
$= k$

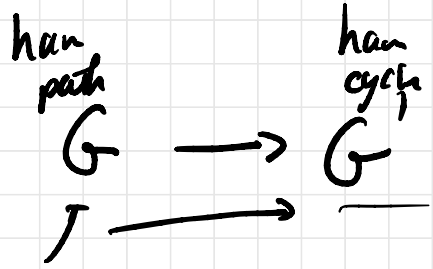
5 Reduction is a P-reduction:

$$\langle G, k \rangle \rightarrow \langle \overline{G}, n-k \rangle$$

adj matrix: flip all 0/1s : $O(n^2)$

$k \rightarrow n-k$ $O(1)$


G no ~~has~~ a ham path
u
s
t



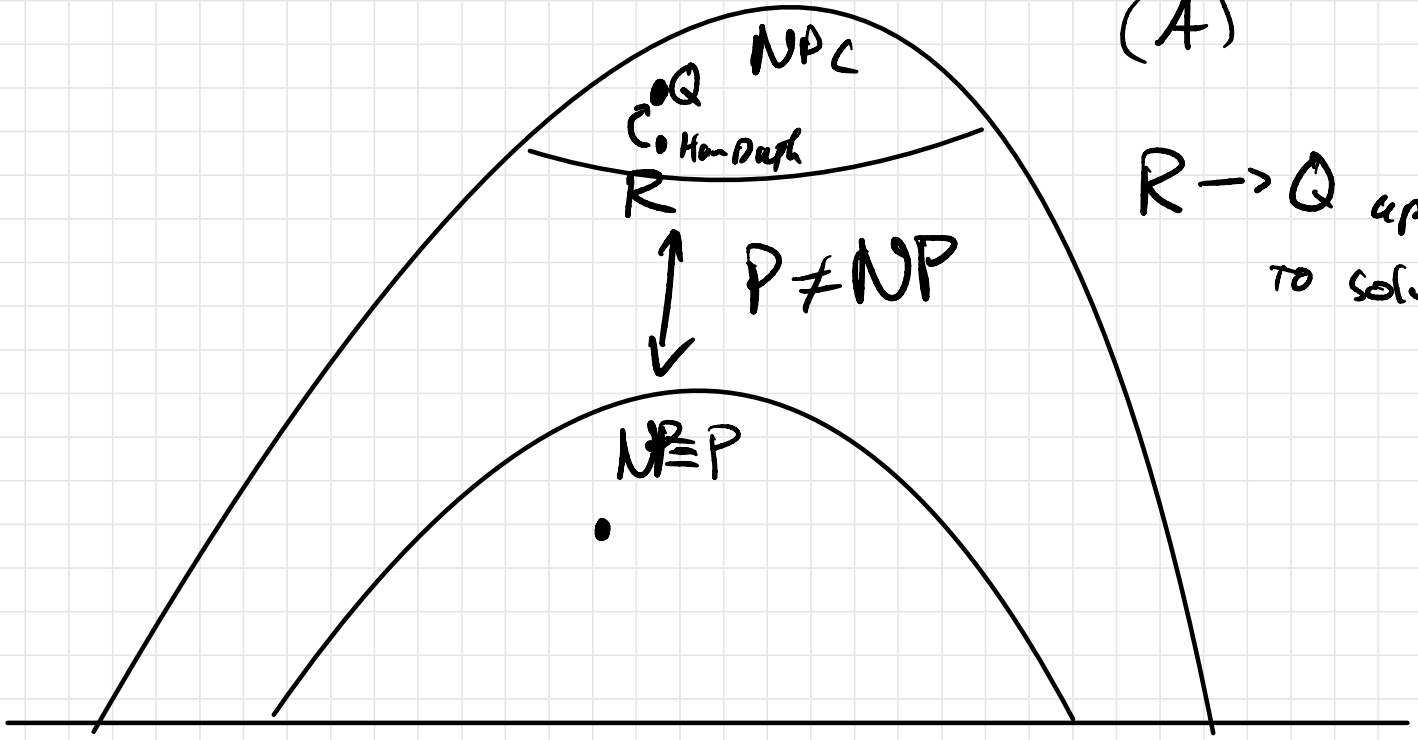
$P = NP$

$A(G')$ if accepts, $\Rightarrow$

G has a ham path

if reject $\Rightarrow$

G no ham path



∃ a poly algo for Q !!

(A)

R → Q applied A to solve R