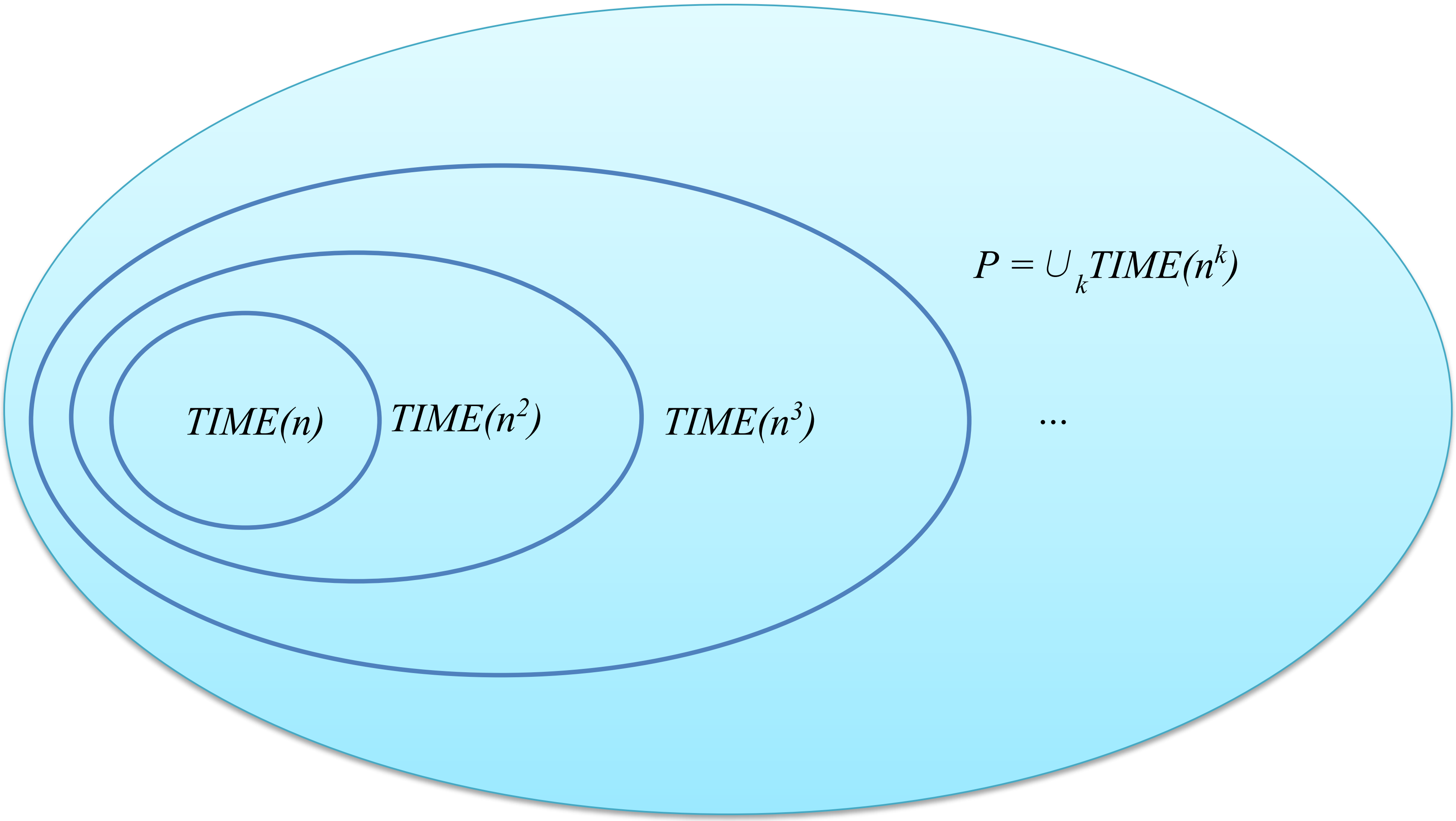
P and NP

Sipser 7.2-7.3 (pages 256-270)

Polynomial time



“Practical” problems

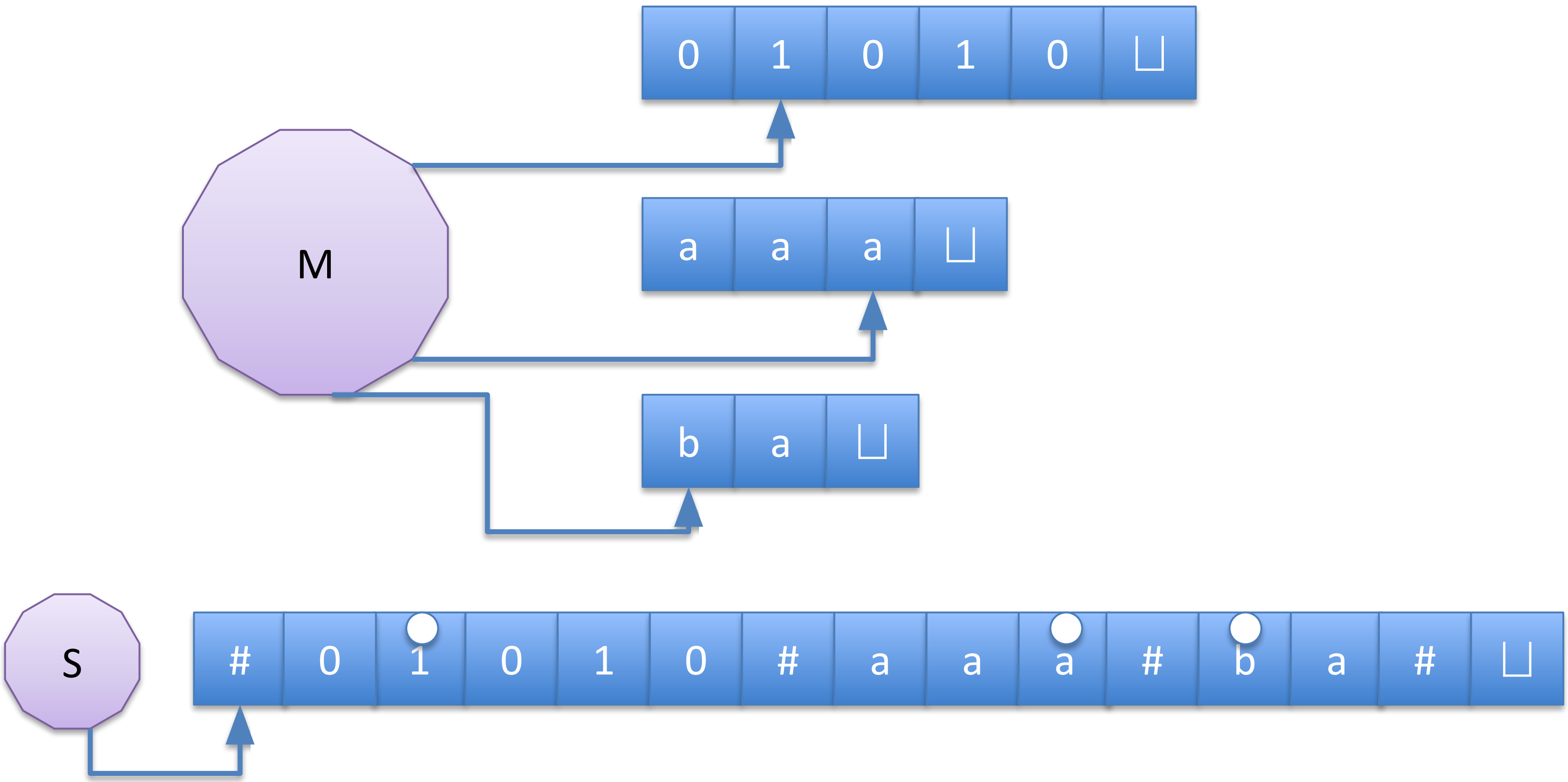
- If $n = 100$
  - $n^3 = 1$ billion
  - $2^n >$ #atoms in the universe
- Polynomial time is generally considered “practical” for a computer
- P is the class of “solvable” or “tractable” problems

How many tapes?

- Definition 7.12:

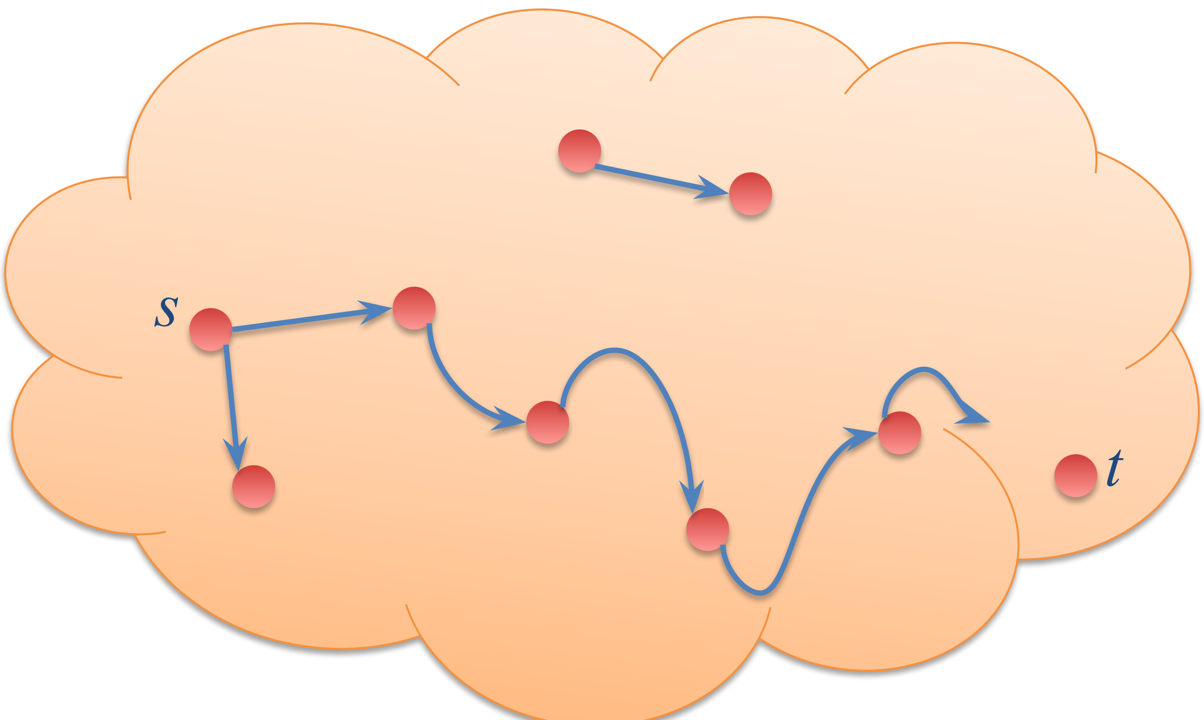
  P is the class of languages that are decidable in polynomial time by a deterministic single-tape Turing machine.
- But remember… we can convert from multi-tape to single-tape!
  - What was the time complexity conversion?

Polynomially equivalent models



Finding your way

- $PATH = \{<G,s,t> \mid \exists$ *directed s to t path in G*$\}$

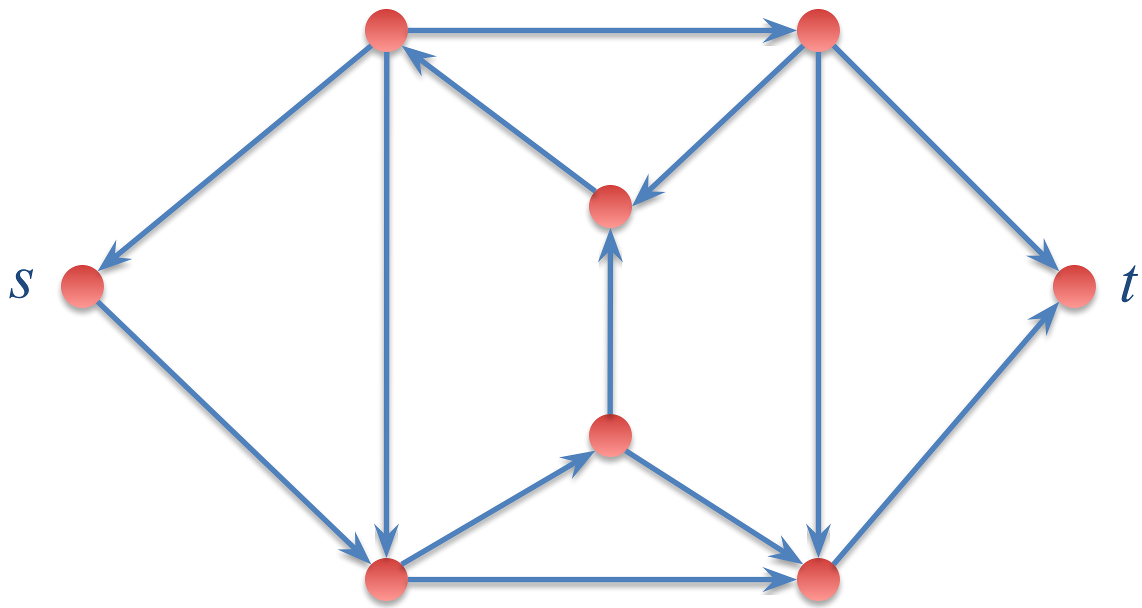


$PATH \in P$

- M = "On input $<G,s,t>$:
  1. Place a mark on node s.
  2. Repeat until no additional nodes are marked:
  3. Scan all edges of G.
  4. If (a,b) found from marked node to unmarked node, mark b.
  5. If t is marked, *accept*. Otherwise, *reject*.

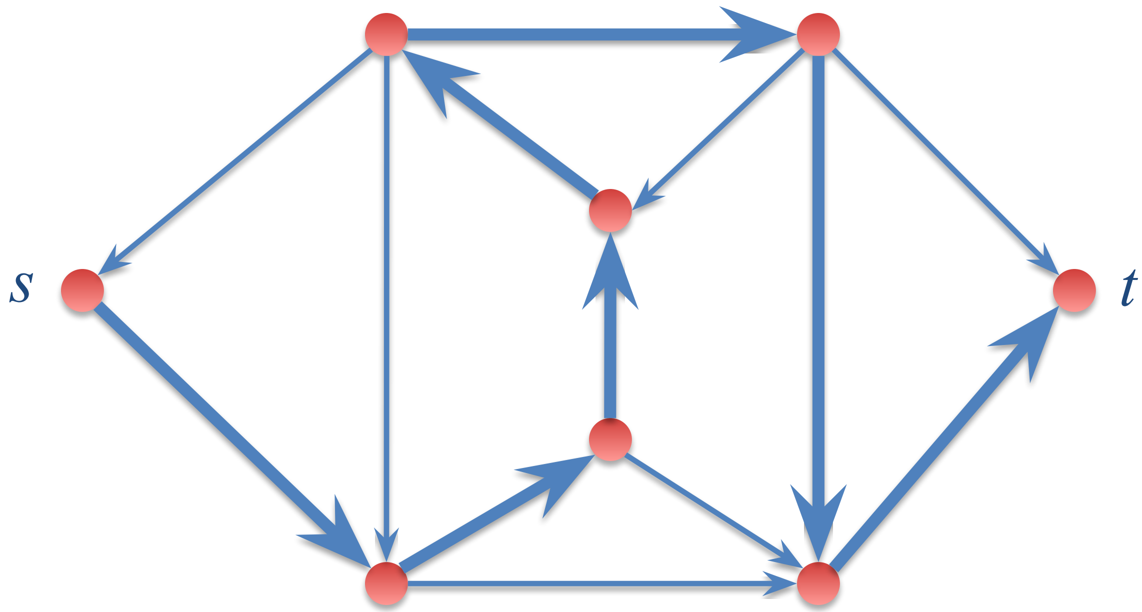
Hamiltonian paths

- $HAMPATH = \{\langle G,s,t \rangle \mid \exists \text{ Hamiltonian path from } s \text{ to } t\}$



Hamiltonian paths

- $HAMPATH = \{\langle G,s,t\rangle \mid \exists$ *Hamiltonian path from* s *to* $t\}$



Checking for Hamiltonian paths

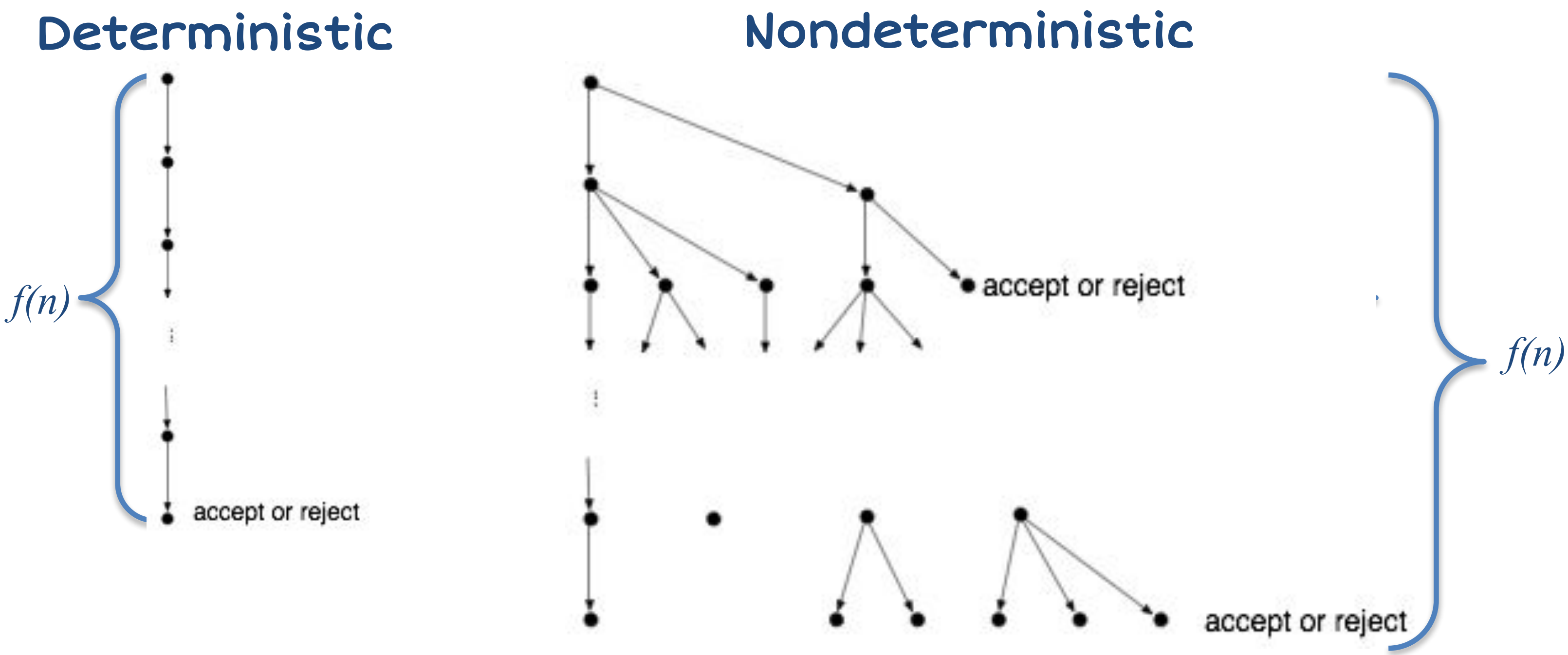
- Brute force method
- E = "On input $<G,s,t>$:
  1. Generate all orderings, $p_1, p_2, \ldots, p_n$, of the nodes of G
  2. For each ordering:
  3. Check whether $s = p_1$ and $t = p_n$
  4. For each $i=1$ to $n-1$, check whether (p_i, p_{i+1}) is an edge of G.
  5. *Accept* if a Hamiltonian path is found.
  6. If no ordering gives a Hamiltonian path, *reject*."

Guessing a solution

- N = "On input <G,s,t>:
  1. Guess an ordering, $p_1, p_2, \ldots, p_n$, of the nodes of G
  2. Check whether $s = p_1$ and $t = p_n$
  3. For each $i=1$ to $n\text{-}1$, check whether (p_i, p_{i+1}) is an edge of G.
  4. If any are not, *reject*. Otherwise, *accept*."

Nondeterministic time complexity

- Definition 7.9: Let N be a NTM. The **running time** of N is a function $f: \mathbf{N} \rightarrow \mathbf{N}$, where $f(n)$ is the maximum number of steps that N uses on any branch of its computation on any input of length n



Nondeterministic time complexity classes

- Definition 7.21:

  $NTIME(t(n)) =$

  $\{L \mid L \text{ is decided in } O(t(n)) \text{ time by an NTM}\}$

Verifiers

- Definition 7.18:
  - A verifier for a language A is an algorithm V, where

    $$A = \{w \mid V \textit{ accepts } <w,c> \textit{ for some string } c\}$$

  - A polynomial time verifier runs in *polynomial* time in the length of w
  - A language A is polynomially verifiable if it has a polynomial time verifier

The class NP

- Definition 7.19:

  NP is the class of languages that have polynomial time verifiers

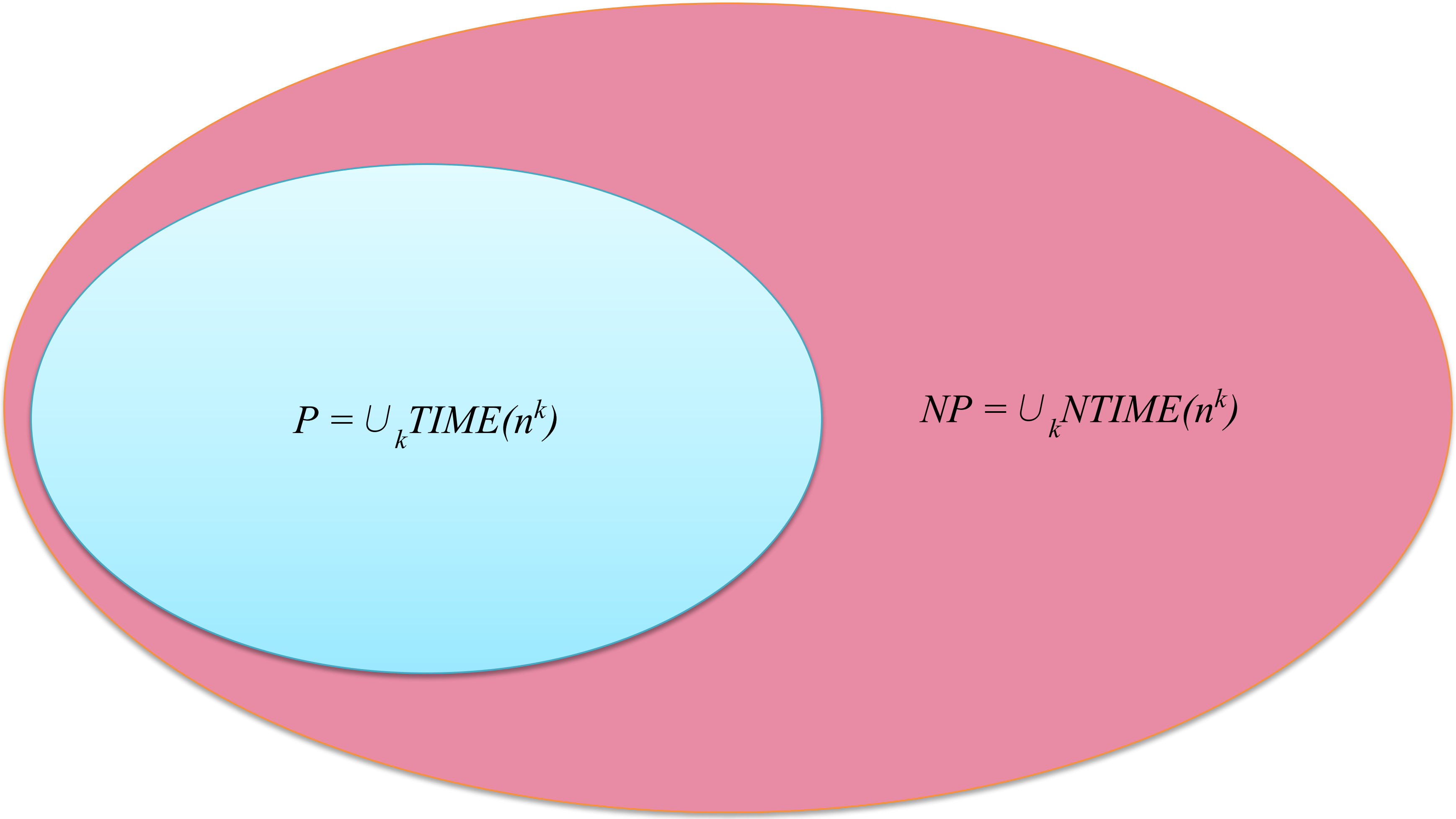
Nondeterminism and verifiers

- Theorem 7.20:

  A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine

- Corollary 7.19: $NP = \bigcup_k NTIME(n^k)$

The classes P and NP



The big question: $P = NP$?

$$\underbrace{P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME}_{\text{proper containment}}$$