LEC19 Complexity

Topday: Computational Complexity

-P,NP,EXP,R
-most problems are uncomputable
-hardness & completeness
-reductions

P

{problems solvable in polynomial time $n^{O(1)}$}
where n = problem/input size

EXP

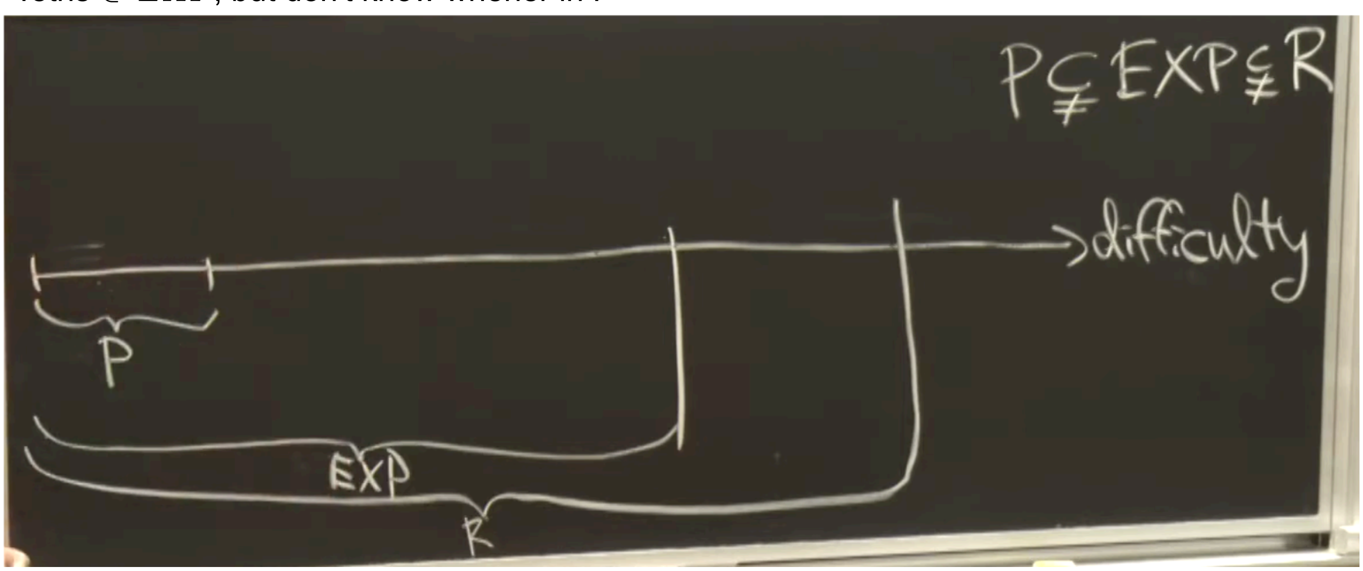
{problems solvable in exponential time $2^{n^{O(1)}}$}

R

{problems solvable in finite time}

Examples:

nxn Chess is EXP
-negative-weight cycle detection $\in\ P$
-Tetris $\in\ EXP$, but don't know wheher in P

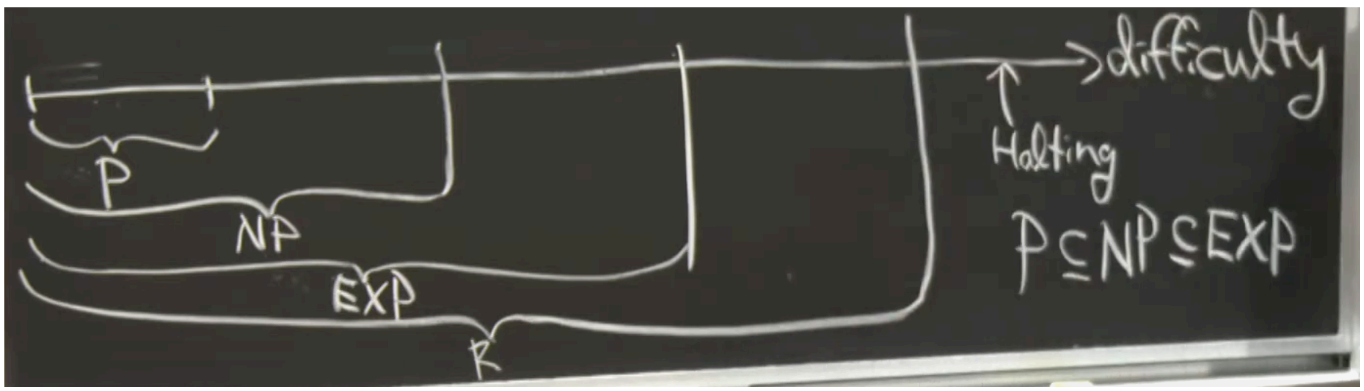


Halting problem:

$\notin R$

- Example
  given a computer program, does it ever halt(stop) ?
  -uncomputable($\notin R$)
- Most decision problems are uncomputable:
  -program ~ finite string of bits ~ number $\in N$
  ->*decision problem* ~ function from inputs(string of bits ~ number in N) ->{YES, NO} ~ infinite string of bits of infinite input [uncountable] ~
  -no assignment of programs -> problems
  -luckily most problem we care about $\in R$

NP



- {decision problems solvable in poly.time via a "lucky" algorithm}
  take a guess that always right
  -nondeterministic model: algorithm can make guesses then output YES or NO
  -guesses guaranteed to lead to YES outcome if possible

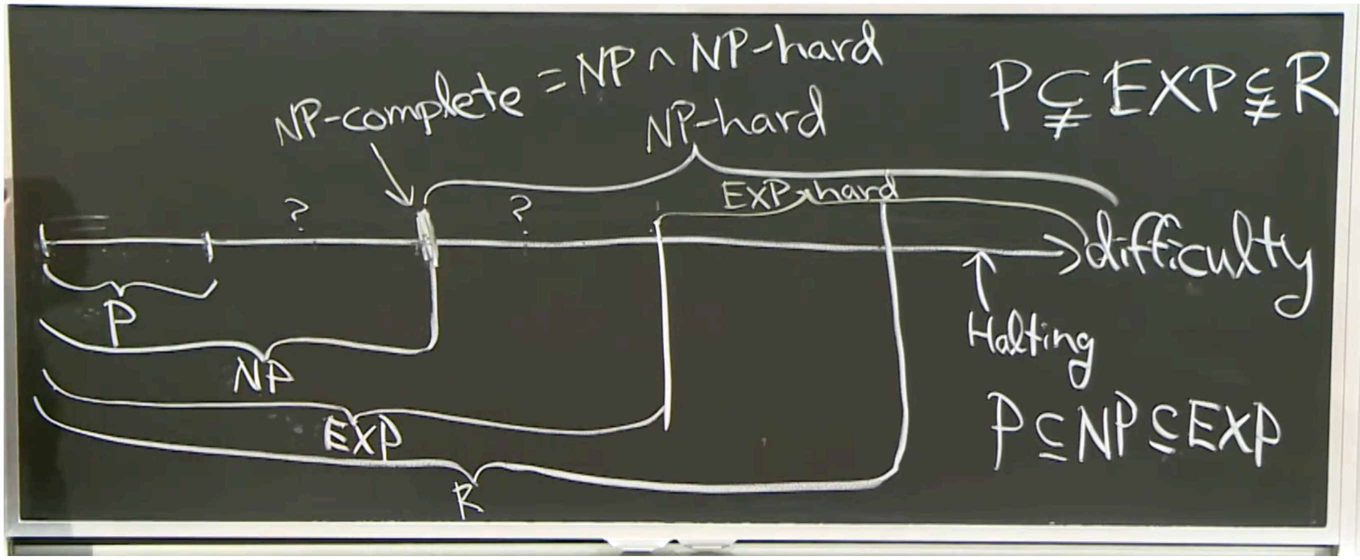
Tetris $\in NP$

俄罗斯方块
version 1
-for each piece. guess how to place
-check rules
-if survive: return YES

- {decision problems with YES solutions that can be cheked in polynomial time}
  version 2
  -certificate for YES input = sequence of moves for pieces
  -given problem input + certificate poly-time verification algorithm
  -for every YES inpute Exist certificate: verfier says YES
  -every no input all vertificate: verfier says NO

P != NP ?

np are problems you can solve by lucky algorithms
p are problems you can solve by regular old algorithms
-Claim:
if P != NP, then Tetris $\notin P$
WHY? Tetris is NP-hard = "as hard as all problems in NP"



Reductions:

A input -> B input ---> B solution -> A solution

ex: unweighted SSSP -> weighted SSSP
longest path -> shotest path

A is at least as easy as B congrant with B is at least as hard as A