# Computational Complexity

Lecture 0

# Computation

# Computation

- A paradigm of modern science

# Computation

- A paradigm of modern science
- Theory of computation/computational complexity is to computer science what theoretical physics is to electronics

# Computational Complexity

# Computational Complexity

- Computation:

# Computational Complexity

- Computation:
  - Problems to be solved

# Computational Complexity

- Computation:
  - Problems to be solved
  - Algorithms to solve them

# Computational Complexity

- Computation:
  - Problems to be solved
  - Algorithms to solve them
    - in various models of computation

# Computational Complexity

- Computation:
  - Problems to be solved
  - Algorithms to solve them
    - in various models of computation
- Complexity of a problem (in a comp. model)

# Computational Complexity

- Computation:
  - Problems to be solved
  - Algorithms to solve them
    - in various models of computation
- Complexity of a problem (in a comp. model)
  - How much "resource" is sufficient/necessary

# Computational Complexity

of Problems

in Models of computation

w.r.t Complexity measures

# Problems

# Problems

- Input represented as (say) a binary string

# Problems

- Input represented as (say) a binary string
- Given input, find a "satisfactory output"

# Problems

- Input represented as (say) a binary string
- Given input, find a "satisfactory output"
  - Function evaluation: only one correct output

# Problems

- Input represented as (say) a binary string
- Given input, find a "satisfactory output"
  - Function evaluation: only one correct output
    - Approximate evaluation

# Problems

- Input represented as (say) a binary string
- Given input, find a "satisfactory output"
  - Function evaluation: only one correct output
    - Approximate evaluation
  - Search problem: find one of many (if any)

# Problems

- Input represented as (say) a binary string
- Given input, find a "satisfactory output"
  - Function evaluation: only one correct output
    - Approximate evaluation
  - Search problem: find one of many (if any)
  - Decision problem: find out if any

# Problems

- Input represented as (say) a binary string
- Given input, find a "satisfactory output"
  - Function evaluation: only one correct output
    - Approximate evaluation
  - Search problem: find one of many (if any)
  - Decision problem: find out if any
    - A Boolean function evaluation (TRUE/FALSE)

# Decision Problems

# Decision Problems

- Evaluate a Boolean function (TRUE/FALSE)

# Decision Problems

- Evaluate a Boolean function (TRUE/FALSE)
  - i.e., Decide if input has some property

# Decision Problems

- Evaluate a Boolean function (TRUE/FALSE)
  - i.e., Decide if input has some property
- Language

# Decision Problems

- Evaluate a Boolean function (TRUE/FALSE)
  - i.e., Decide if input has some property
- Language
  - Set of inputs with a particular property

# Decision Problems

- Evaluate a Boolean function (TRUE/FALSE)
  - i.e., Decide if input has some property
- Language
  - Set of inputs with a particular property
  - e.g. L = {x | x has equal number of 0s and 1s}

# Decision Problems

- Evaluate a Boolean function (TRUE/FALSE)
  - i.e., Decide if input has some property
- Language
  - Set of inputs with a particular property
  - e.g. L = {x | x has equal number of 0s and 1s}
  - Decision problem: given input, decide if it is in L

# Complexity of Languages

# Complexity of Languages

- Some languages are "simpler" than others

# Complexity of Languages

- Some languages are "simpler" than others
  - $L_1$ = {x | x starts with 0}

# Complexity of Languages

- Some languages are "simpler" than others
  - $L_1$ = {x | x starts with 0}
  - $L_2$ = {x | x has equal number of 0s and 1s}

# Complexity of Languages

- Some languages are "simpler" than others
  - $L_1$ = {x | x starts with 0}
  - $L_2$ = {x | x has equal number of 0s and 1s}
- Simpler in what way?

# Complexity of Languages

- Some languages are "simpler" than others
  - $L_1$ = {x | x starts with 0}
  - $L_2$ = {x | x has equal number of 0s and 1s}
- Simpler in what way?
  - Fewer calculations, less memory, need not read all input, can do in an FSM

# Complexity of Languages

- Some languages are "simpler" than others
  - $L_1$ = {x | x starts with 0}
  - $L_2$ = {x | x has equal number of 0s and 1s}
- Simpler in what way?
  - Fewer calculations, less memory, need not read all input, can do in an FSM

# Complexity of Languages

# Complexity of Languages

- Relating complexities of problems

# Complexity of Languages

- Relating complexities of problems
  - Mo = {x | x has more 0s than 1s}

# Complexity of Languages

- Relating complexities of problems
  - Mo = {x | x has more 0s than 1s}
  - Eq = {x | x has equal number of 0s and 1s}

# Complexity of Languages

- Relating complexities of problems
  - Mo = {x | x has more 0s than 1s}
  - Eq = {x | x has equal number of 0s and 1s}
  - Eq(x):

# Complexity of Languages

- Relating complexities of problems
  - Mo = {x | x has more 0s than 1s}
  - Eq = {x | x has equal number of 0s and 1s}
  - Eq(x):
    - if (Mo(x0) == TRUE and Mo(x) == FALSE) then TRUE; else FALSE

# Complexity of Languages

- Relating complexities of problems
  - Mo = {x | x has more 0s than 1s}
  - Eq = {x | x has equal number of 0s and 1s}
  - Eq(x):
    - if (Mo(x0) == TRUE and Mo(x) == FALSE) then TRUE; else FALSE
- Eq is not (much) more complex than Mo.
  i.e., Mo is at least (almost) as complex as Eq.

# Complexity of Languages

- Relating complexities of problems
  - Mo = {x | x has more 0s than 1s}
  - Eq = {x | x has equal number of 0s and 1s}
  - Eq(x):
    - if (Mo(x0) == TRUE and Mo(x) == FALSE) then TRUE; else FALSE

Eq reduces to Mo

- Eq is not (much) more complex than Mo.
  i.e., Mo is at least (almost) as complex as Eq.

# Models of Computation

# Models of Computation

- FSM, PDA, TM

# Models of Computation

- FSM, PDA, TM
- Variations: Non-deterministic, probabilistic.
  Other models: quantum computation

# Models of Computation

- FSM, PDA, TM
- Variations: Non-deterministic, probabilistic. Other models: quantum computation
- Church-Turing thesis: TM is as "powerful" as it gets

# Models of Computation

- FSM, PDA, TM
- Variations: Non-deterministic, probabilistic. Other models: quantum computation
- Church-Turing thesis: TM is as "powerful" as it gets
- Not enough TMs (algorithms/programs) to solve all decision problems!

# Models of Computation

- FSM, PDA, TM
- Variations: Non-deterministic, probabilistic. Other models: quantum computation
- Church-Turing thesis: TM is as "powerful" as it gets
- Not enough TMs (algorithms/programs) to solve all decision problems!
- Non-uniform computation: circuit families

# Complexity Measures

# Complexity Measures

- Number of computational steps, amount of memory, circuit size/depth, ...

# Complexity Measures

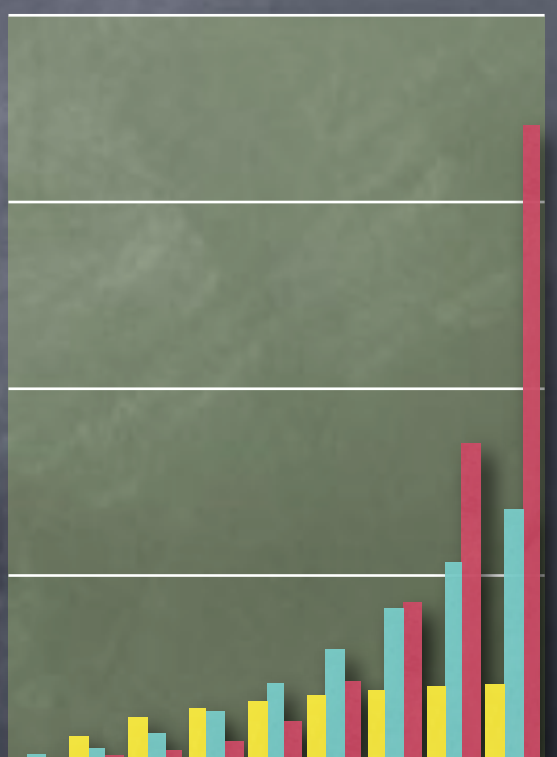
- Number of computational steps, amount of memory, circuit size/depth, ...
- Exact numbers very much dependent on exact specification of the model (e.g. no. of tapes in TM)

# Complexity Measures

- Number of computational steps, amount of memory, circuit size/depth, ...
- Exact numbers very much dependent on exact specification of the model (e.g. no. of tapes in TM)
  - But "broad trends" robust

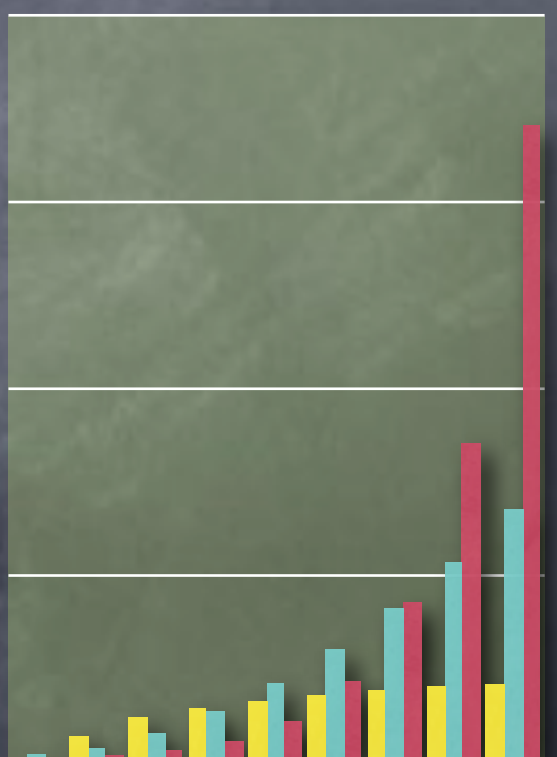
# Complexity Measures

- Number of computational steps, amount of memory, circuit size/depth, ...
- Exact numbers very much dependent on exact specification of the model (e.g. no. of tapes in TM)
  - But "broad trends" robust



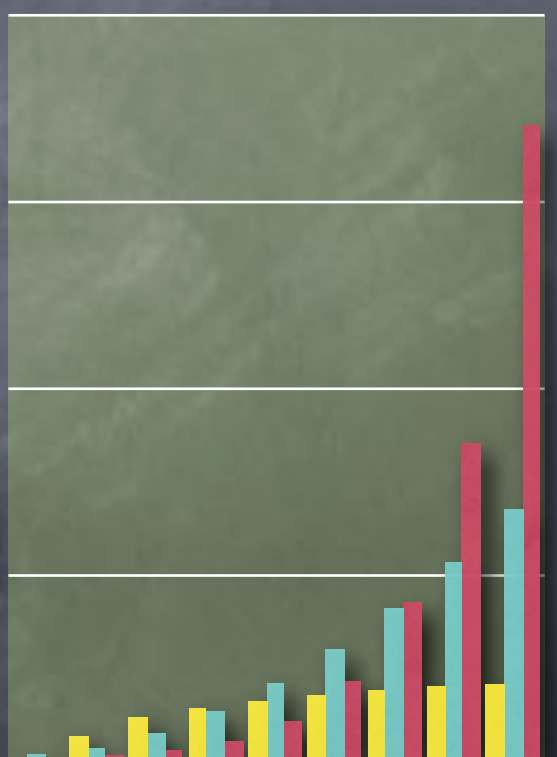
# Complexity Measures

- Number of computational steps, amount of memory, circuit size/depth, ...
- Exact numbers very much dependent on exact specification of the model (e.g. no. of tapes in TM)
  - But "broad trends" robust
    - Trends: asymptotic



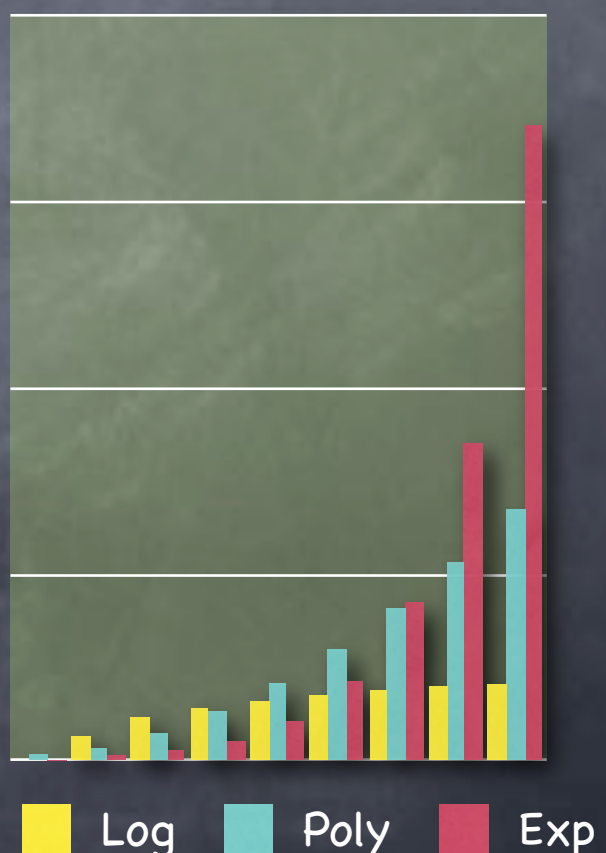
# Complexity Measures

- Number of computational steps, amount of memory, circuit size/depth, ...
- Exact numbers very much dependent on exact specification of the model (e.g. no. of tapes in TM)
  - But "broad trends" robust
    - Trends: asymptotic
    - Broad: Log, Poly, Exp



# Complexity Measures

- Number of computational steps, amount of memory, circuit size/depth, ...
- Exact numbers very much dependent on exact specification of the model (e.g. no. of tapes in TM)
  - But "broad trends" robust
    - Trends: asymptotic
    - Broad: Log, Poly, Exp



# Complexity Theory

# Complexity Theory

- Understand complexity of problems (i.e., how much resource used by best algorithm for it)

# Complexity Theory

- Understand complexity of problems (i.e., how much resource used by best algorithm for it)
  - Relate problems to each other [Reduce]

# Complexity Theory

- Understand complexity of problems (i.e., how much resource used by best algorithm for it)
  - Relate problems to each other [Reduce]
  - Relate computational models/complexity measures to each other [Simulate]

# Complexity Theory

- Understand complexity of problems (i.e., how much resource used by best algorithm for it)
  - Relate problems to each other [Reduce]
  - Relate computational models/complexity measures to each other [Simulate]
  - Calculate complexity of problems

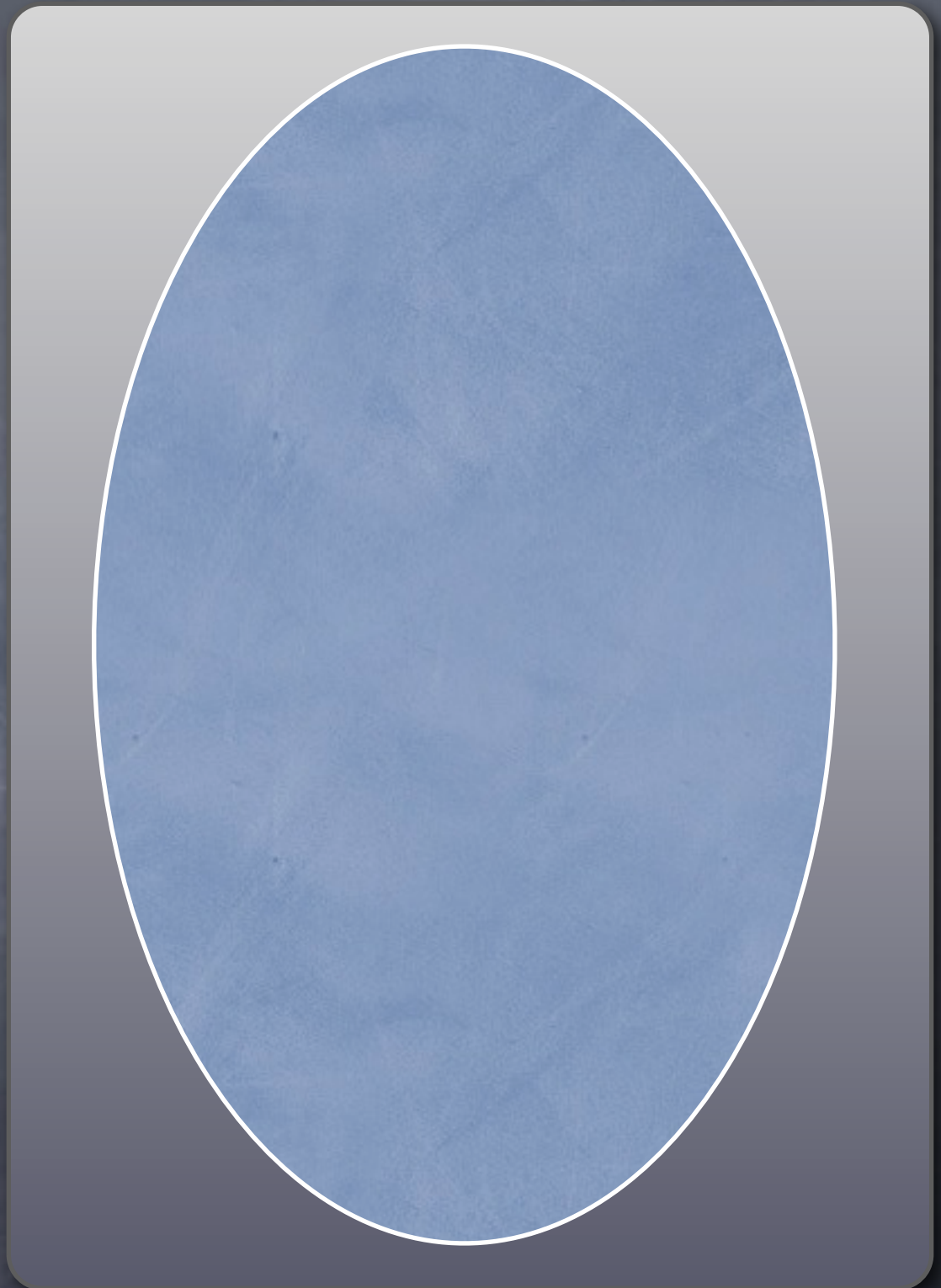
# Complexity Classes

# Complexity Classes

- Collect (decision) problems with similar complexity into classes

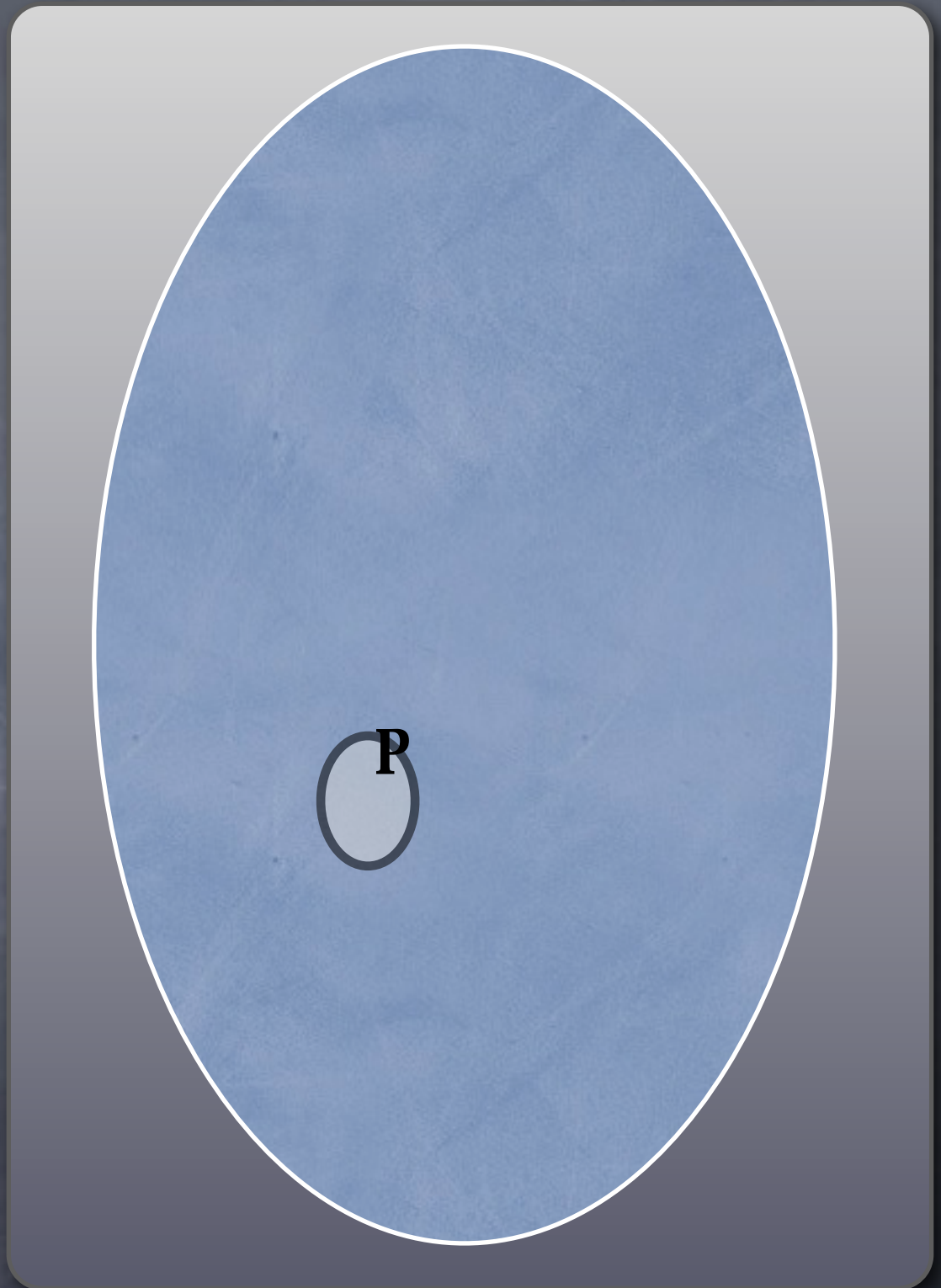
# Complexity Classes

- Collect (decision) problems with similar complexity into classes



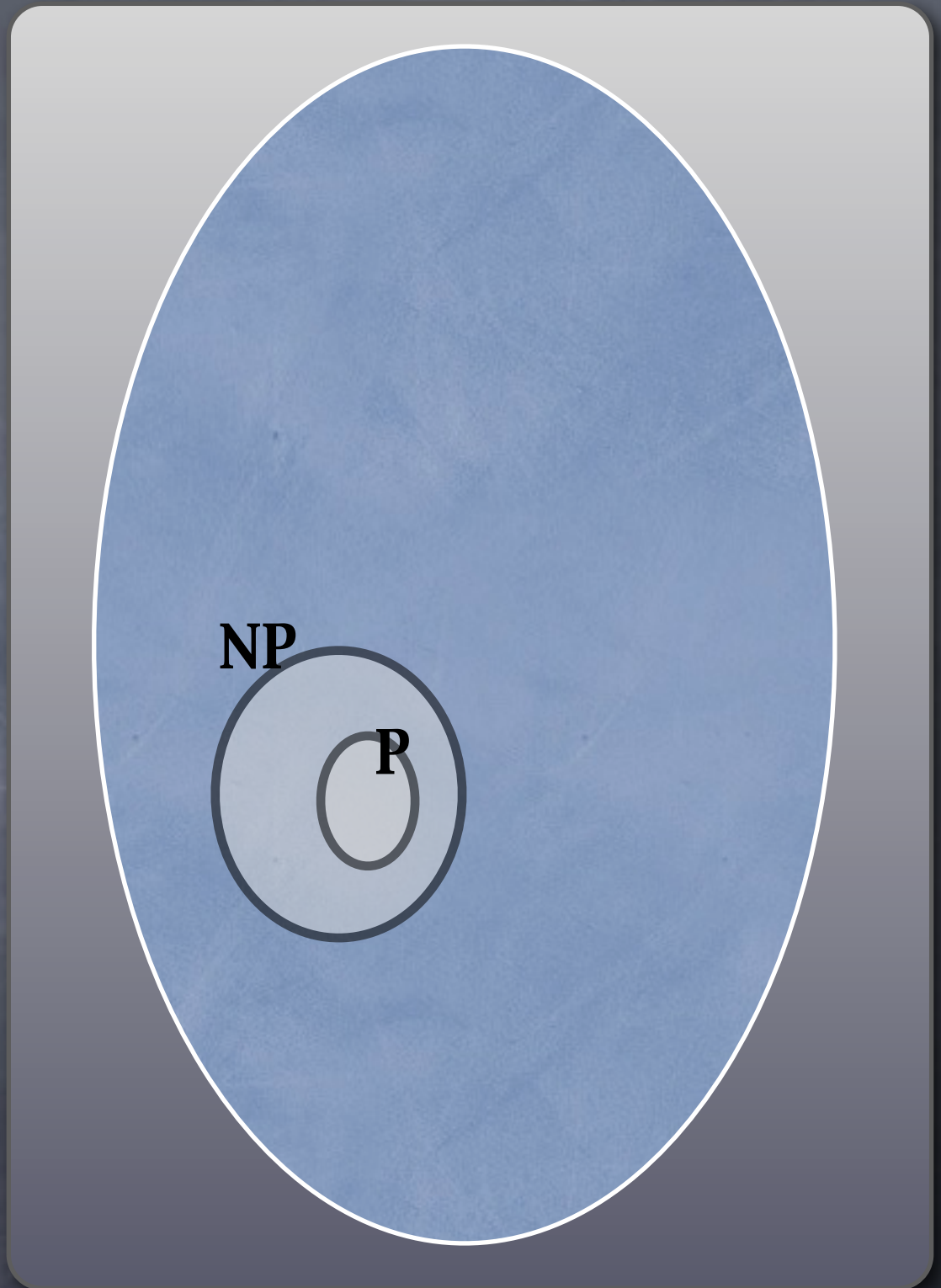
# Complexity Classes

- Collect (decision) problems with similar complexity into classes



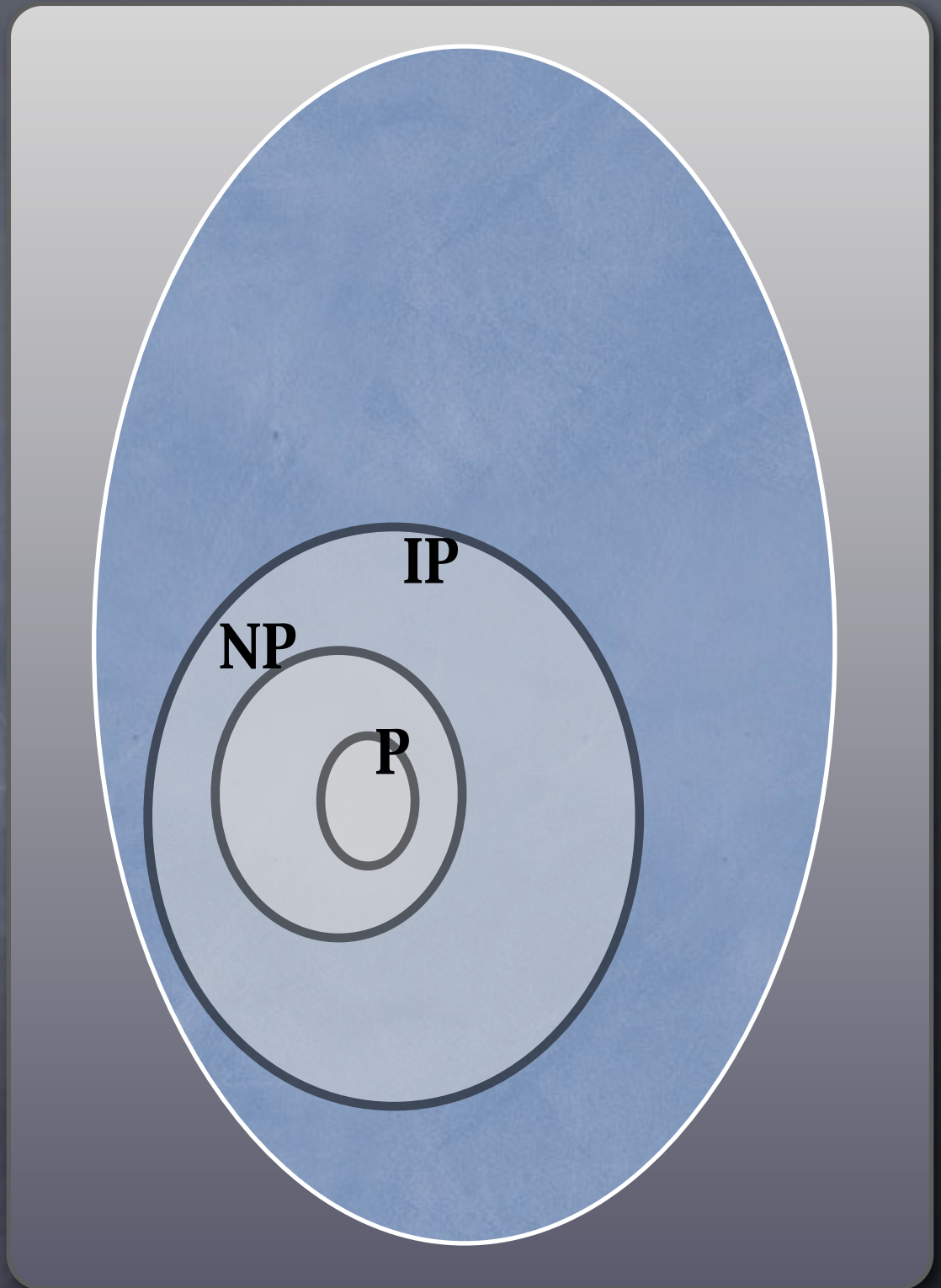
# Complexity Classes

- Collect (decision) problems with similar complexity into classes



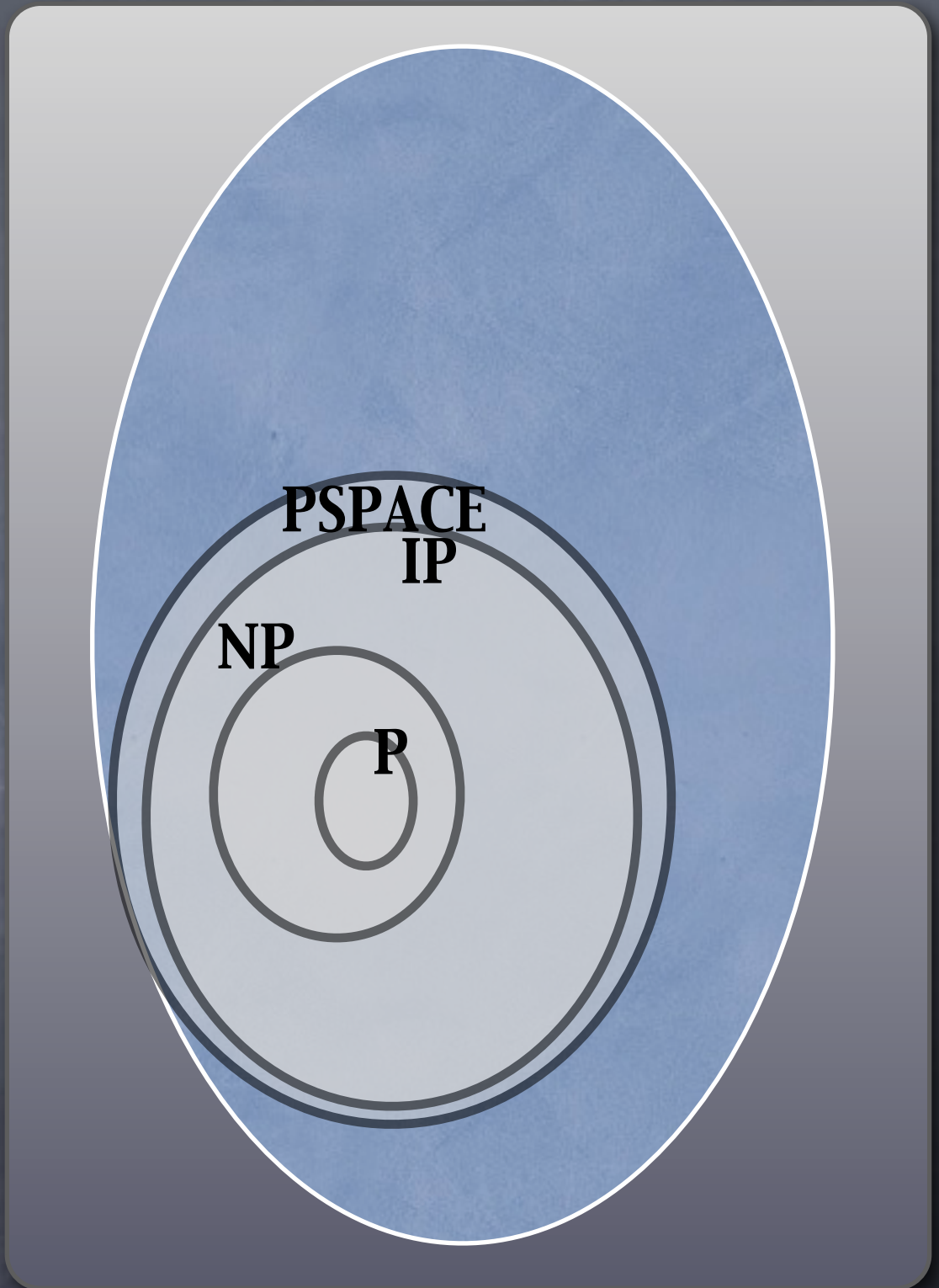
# Complexity Classes

- Collect (decision) problems with similar complexity into classes



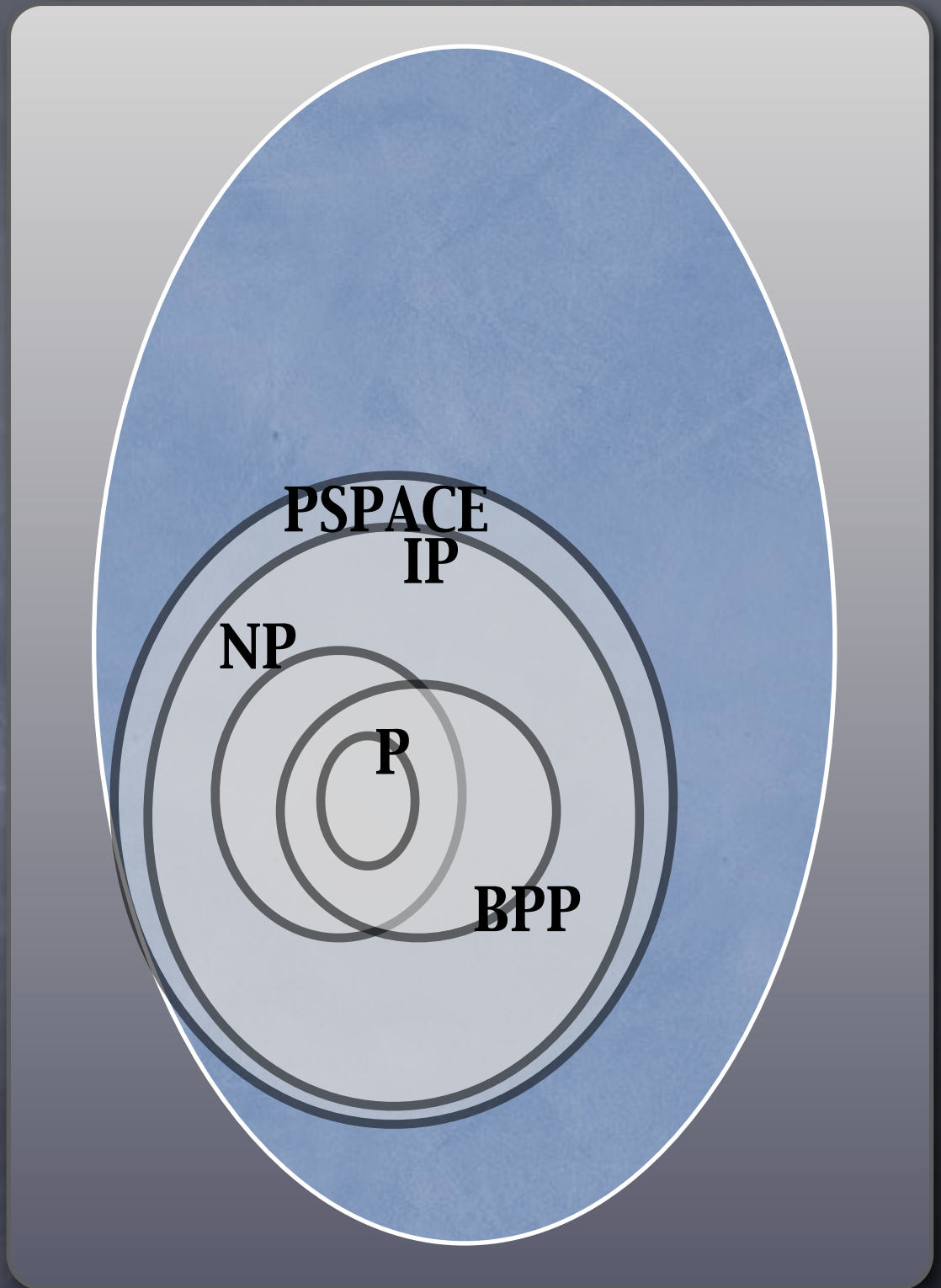
# Complexity Classes

- Collect (decision) problems with similar complexity into classes



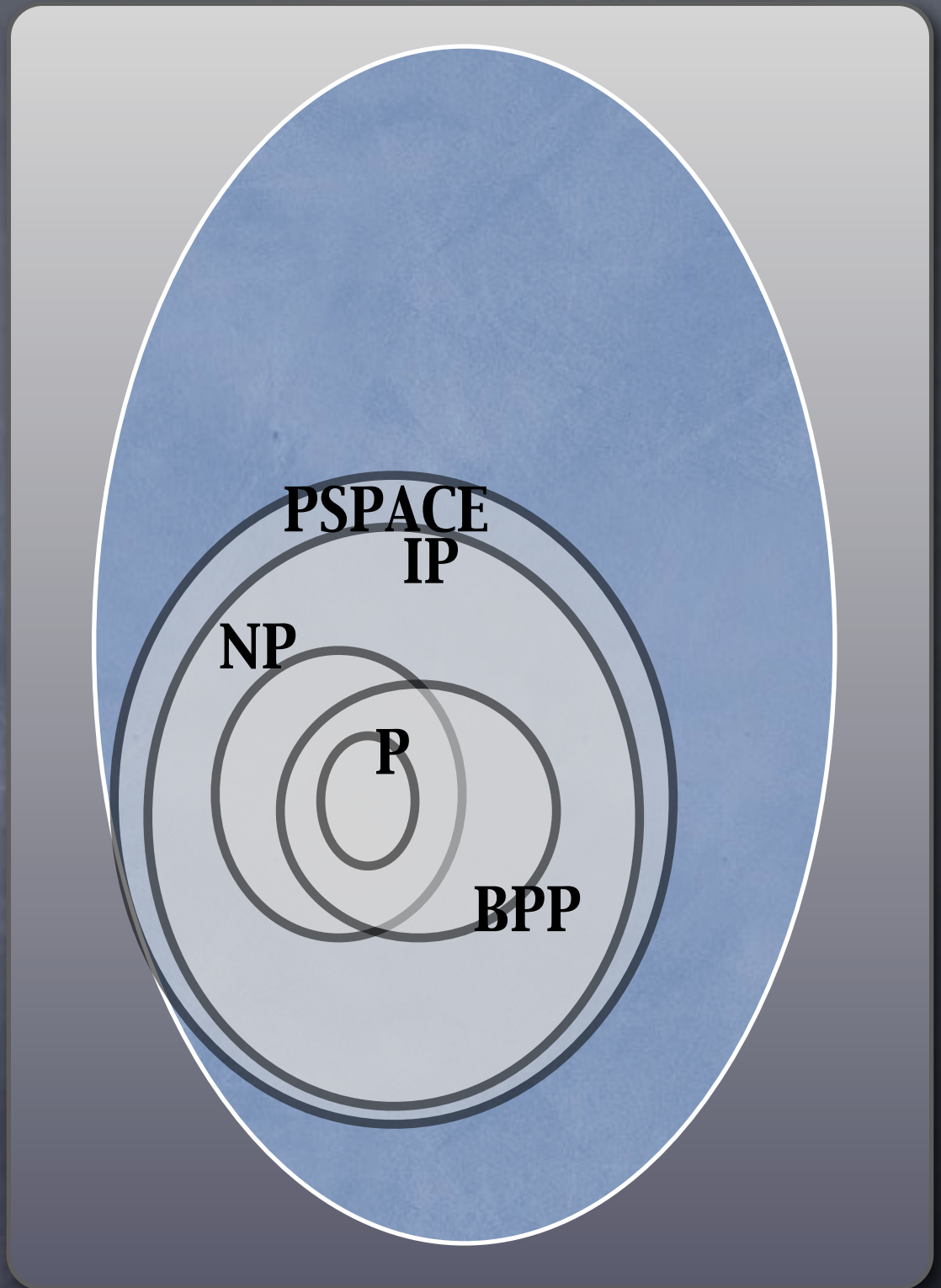
# Complexity Classes

- Collect (decision) problems with similar complexity into classes



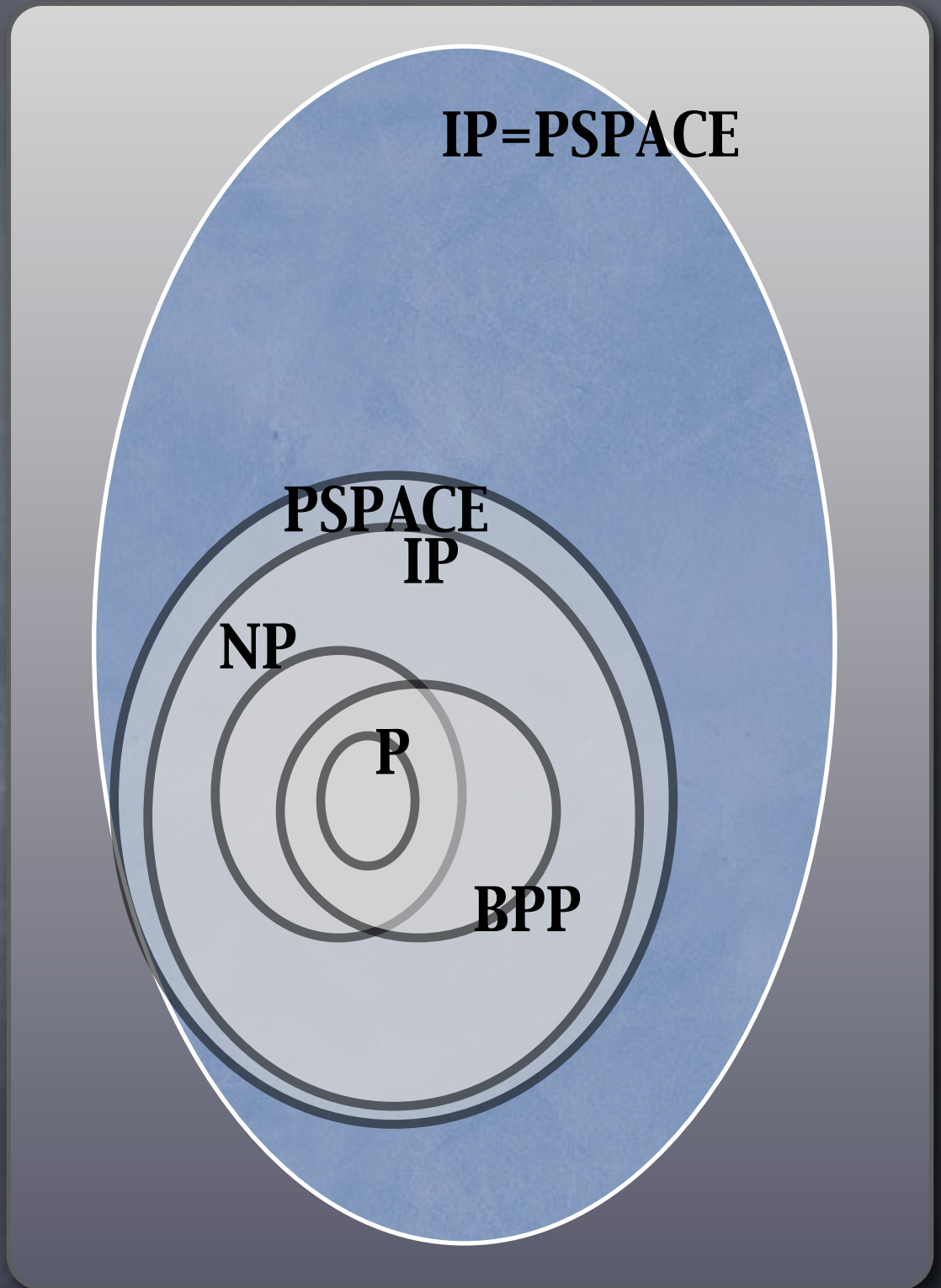
# Complexity Classes

- Collect (decision) problems with similar complexity into classes
  - Relate classes to each other



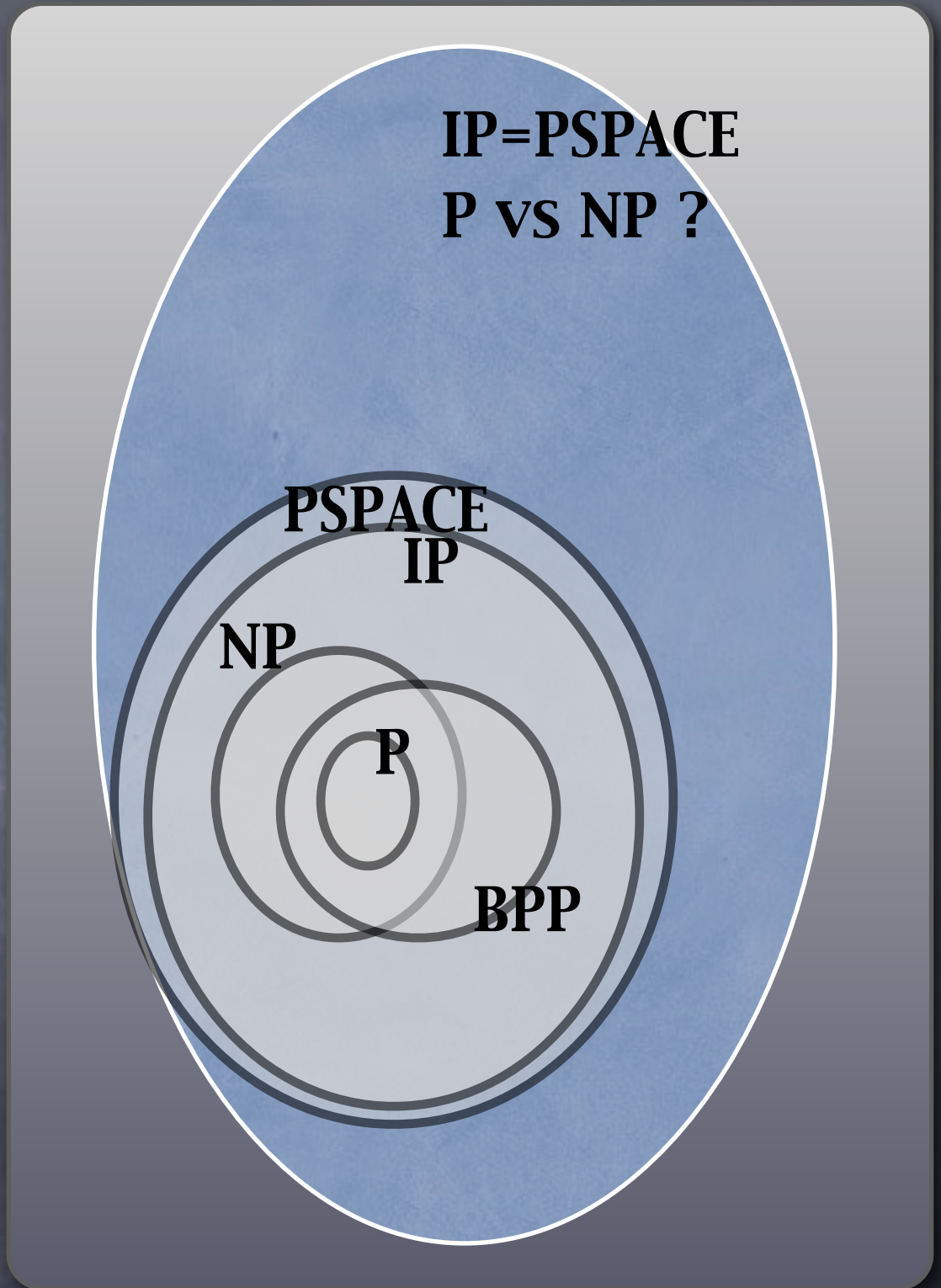
# Complexity Classes

- Collect (decision) problems with similar complexity into classes
  - Relate classes to each other



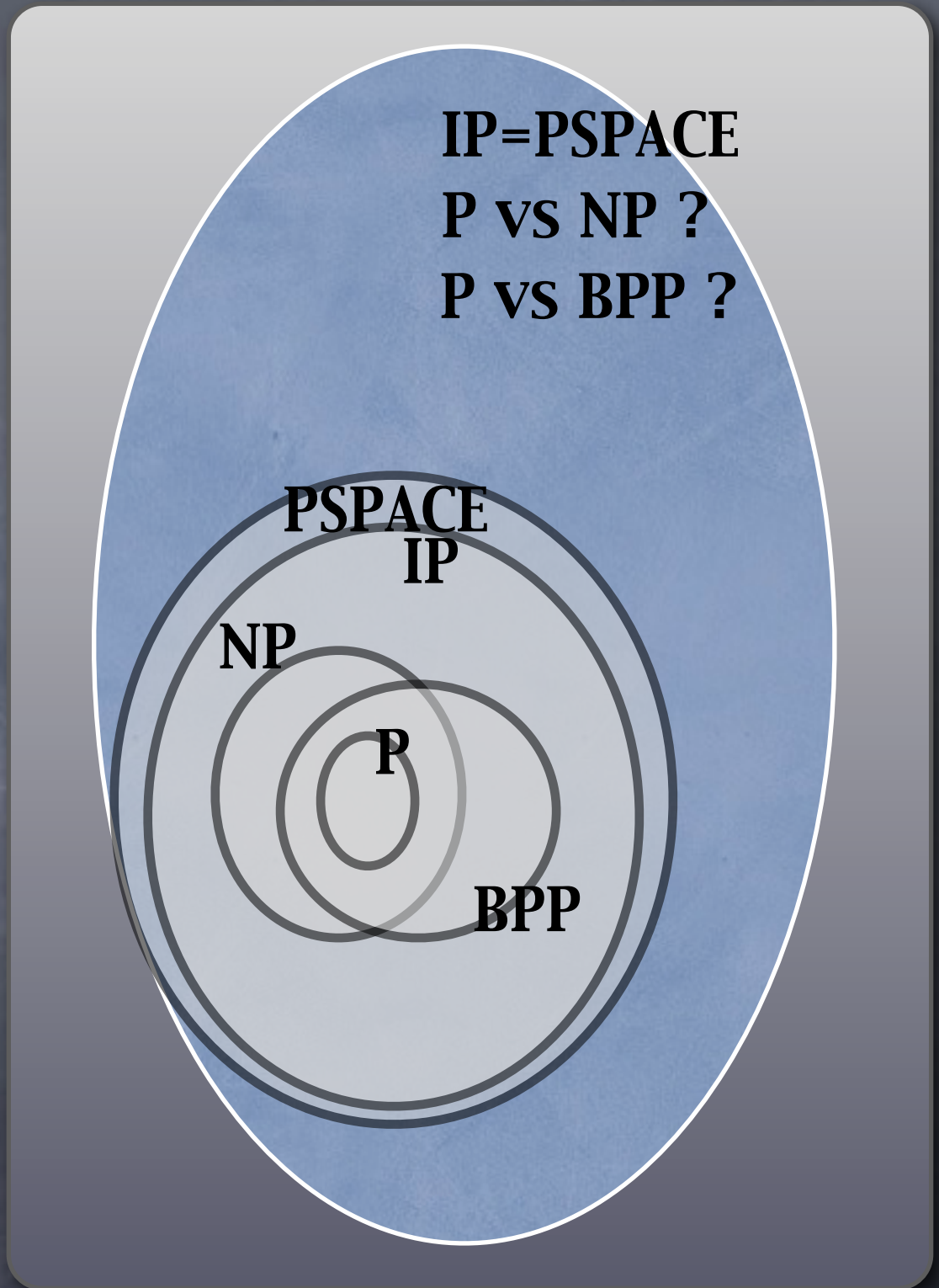
# Complexity Classes

- Collect (decision) problems with similar complexity into classes
  - Relate classes to each other



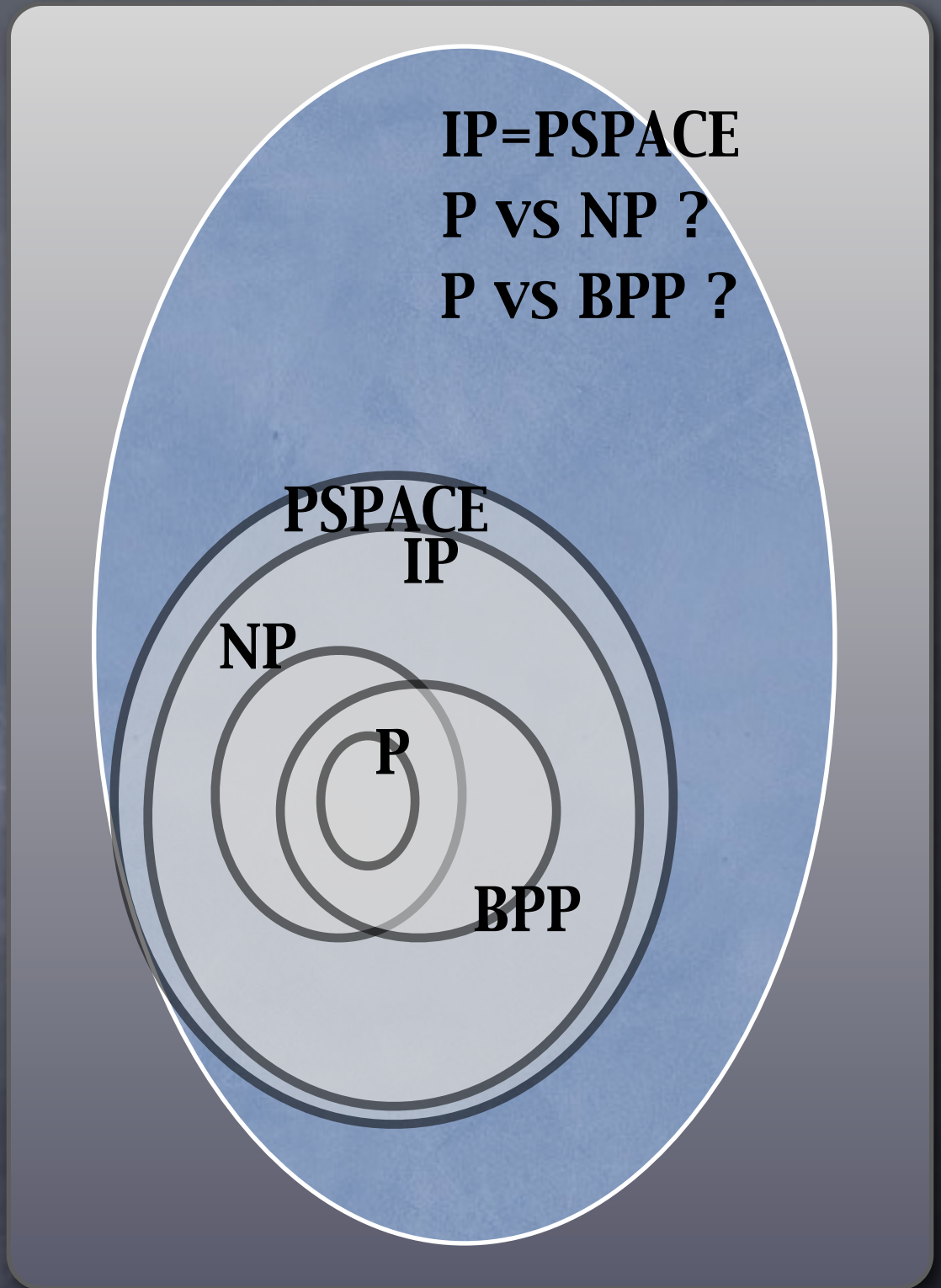
# Complexity Classes

- Collect (decision) problems with similar complexity into classes
  - Relate classes to each other



# Complexity Classes

- Collect (decision) problems with similar complexity into classes
  - Relate classes to each other
  - Hundreds of classes!

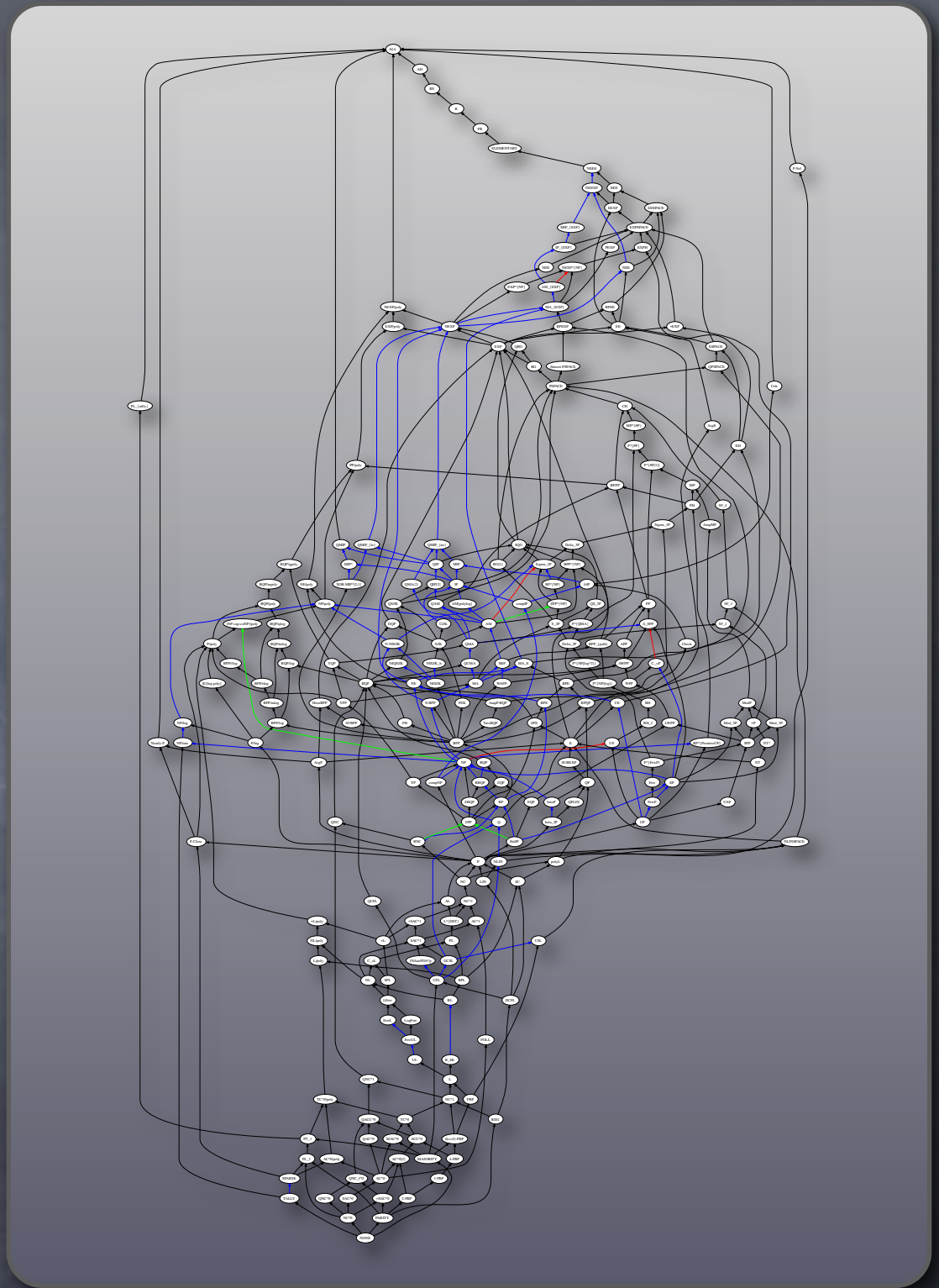


# Complexity Zoo!

- Collect (decision) problems with similar complexity into classes
  - Relate classes to each other
  - Hundreds of classes!

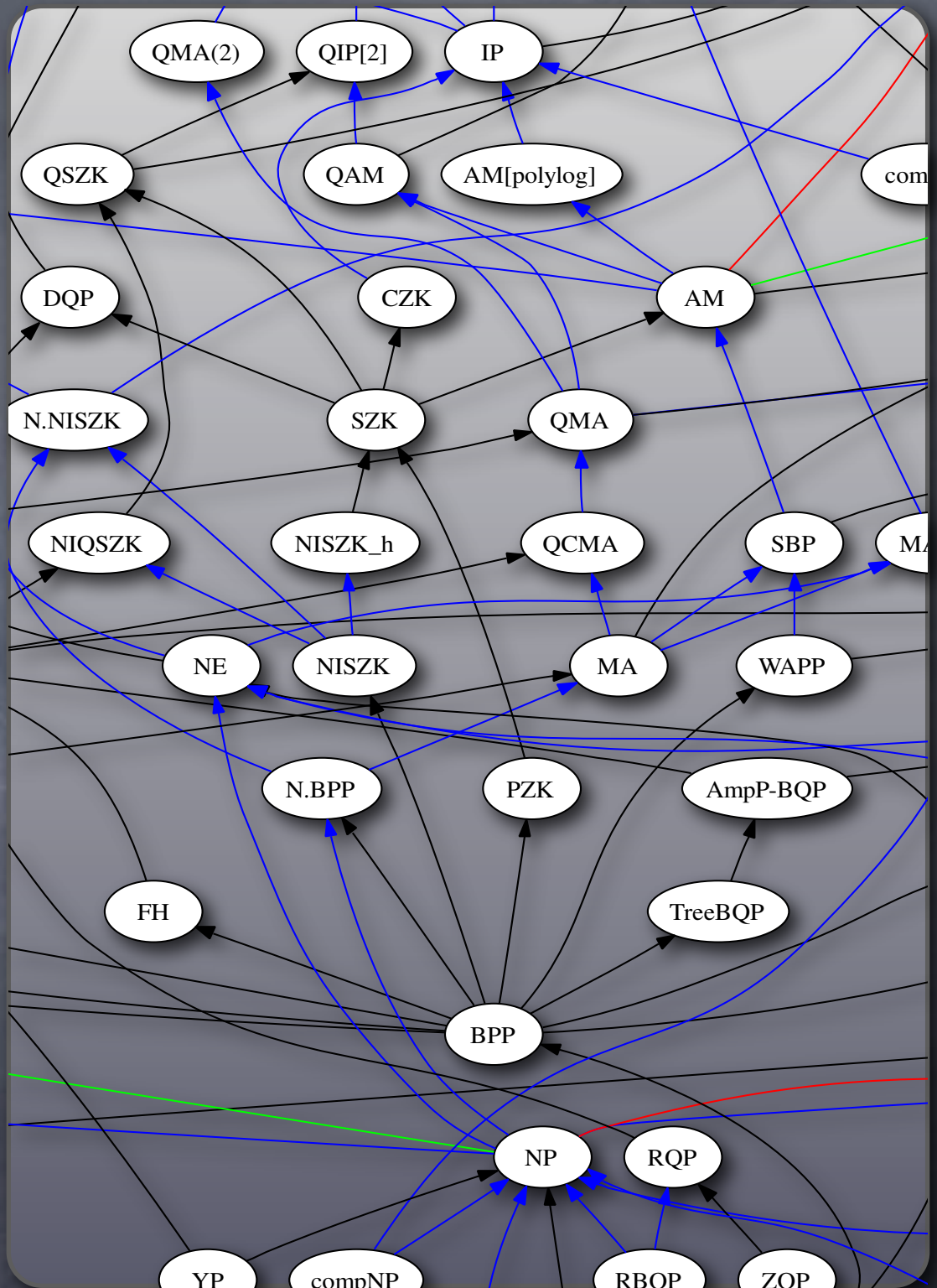
# Complexity Zoo!

- Collect (decision) problems with similar complexity into classes
  - Relate classes to each other
  - Hundreds of classes!



# Complexity Zoo!

- Collect (decision) problems with similar complexity into classes
  - Relate classes to each other
  - Hundreds of classes!



# Complexity in various settings

# Complexity in various settings

- With various models of computation: decision trees, interactive settings, probabilistic computation

# Complexity in various settings

- With various models of computation: decision trees, interactive settings, probabilistic computation
- Various measures: depth, width, amount of communication, number of rounds, amount of randomness, amount of non-uniformity, ...

# Complexity in various settings

- With various models of computation: decision trees, interactive settings, probabilistic computation
- Various measures: depth, width, amount of communication, number of rounds, amount of randomness, amount of non-uniformity, ...
- Various connections: time vs. space, randomness vs. hardness

# Cryptography

# Cryptography

- Need to prove that a scheme is secure (according to some definition)

# Cryptography

- Need to prove that a scheme is secure (according to some definition)
  - i.e., breaking security has high complexity

# Cryptography

- Need to prove that a scheme is secure (according to some definition)
  - i.e., breaking security has high complexity
  - Reductions: if you could break my scheme's security efficiently, I can solve a hard problem almost as efficiently

# Cryptography

- Need to prove that a scheme is secure (according to some definition)
  - i.e., breaking security has high complexity
  - Reductions: if you could break my scheme's security efficiently, I can solve a hard problem almost as efficiently
  - Hard problems: almost all instances hard

# Cryptography

- Need to prove that a scheme is secure (according to some definition)
  - i.e., breaking security has high complexity
  - Reductions: if you could break my scheme's security efficiently, I can solve a hard problem almost as efficiently
  - Hard problems: almost all instances hard
    - For most keys scheme should be secure

# Cryptography

- Need to prove that a scheme is secure (according to some definition)
  - i.e., breaking security has high complexity
  - Reductions: if you could break my scheme's security efficiently, I can solve a hard problem almost as efficiently
  - Hard problems: almost all instances hard
    - For most keys scheme should be secure
- Contrast with Information Theory: limited cryptography if no computational constraints on adversary

# Computational Complexity in Economics/Games

# Computational Complexity in Economics/Games

- Traditionally, no computational constraints for strategizing players

# Computational Complexity in Economics/Games

- Traditionally, no computational constraints for strategizing players
- But many problems in game-theory are computationally hard

# Computational Complexity in Economics/Games

- Traditionally, no computational constraints for strategizing players
- But many problems in game-theory are computationally hard
  - Rational players may not be able to play optimally

# Computational Complexity in Economics/Games

- Traditionally, no computational constraints for strategizing players
- But many problems in game-theory are computationally hard
  - Rational players may not be able to play optimally
  - Suggests the need to redesign markets/financial instruments to be computationally tractable

# Computational Complexity in Economics/Games

- Traditionally, no computational constraints for strategizing players
- But many problems in game-theory are computationally hard
  - Rational players may not be able to play optimally
  - Suggests the need to redesign markets/financial instruments to be computationally tractable
- Recent results...

# All that and much more..

# All that and much more..

- Welcome to CS 579!

# All that and much more..

- Welcome to CS 579!
- Textbook: www.cs.princeton.edu/theory/complexity/

# All that and much more..

- Welcome to CS 579!
- Textbook: www.cs.princeton.edu/theory/complexity/
- About 5 assignments, a class test, and a reading project

# All that and much more..

- Welcome to CS 579!
- Textbook: www.cs.princeton.edu/theory/complexity/
- About 5 assignments, a class test, and a reading project
- Office hours: Wednesdays 2:00 to 3:00 (tentative)