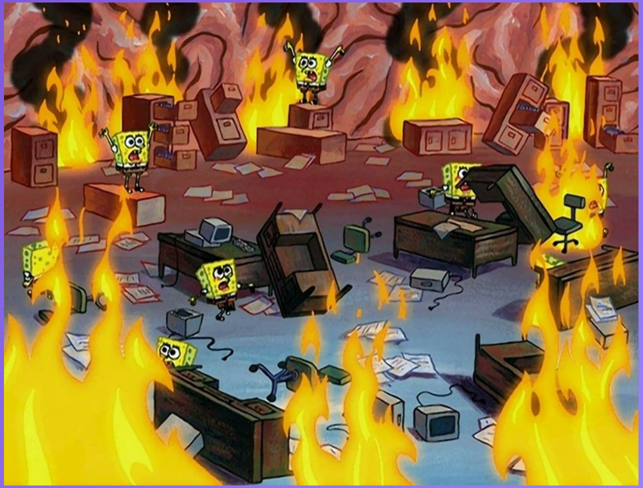
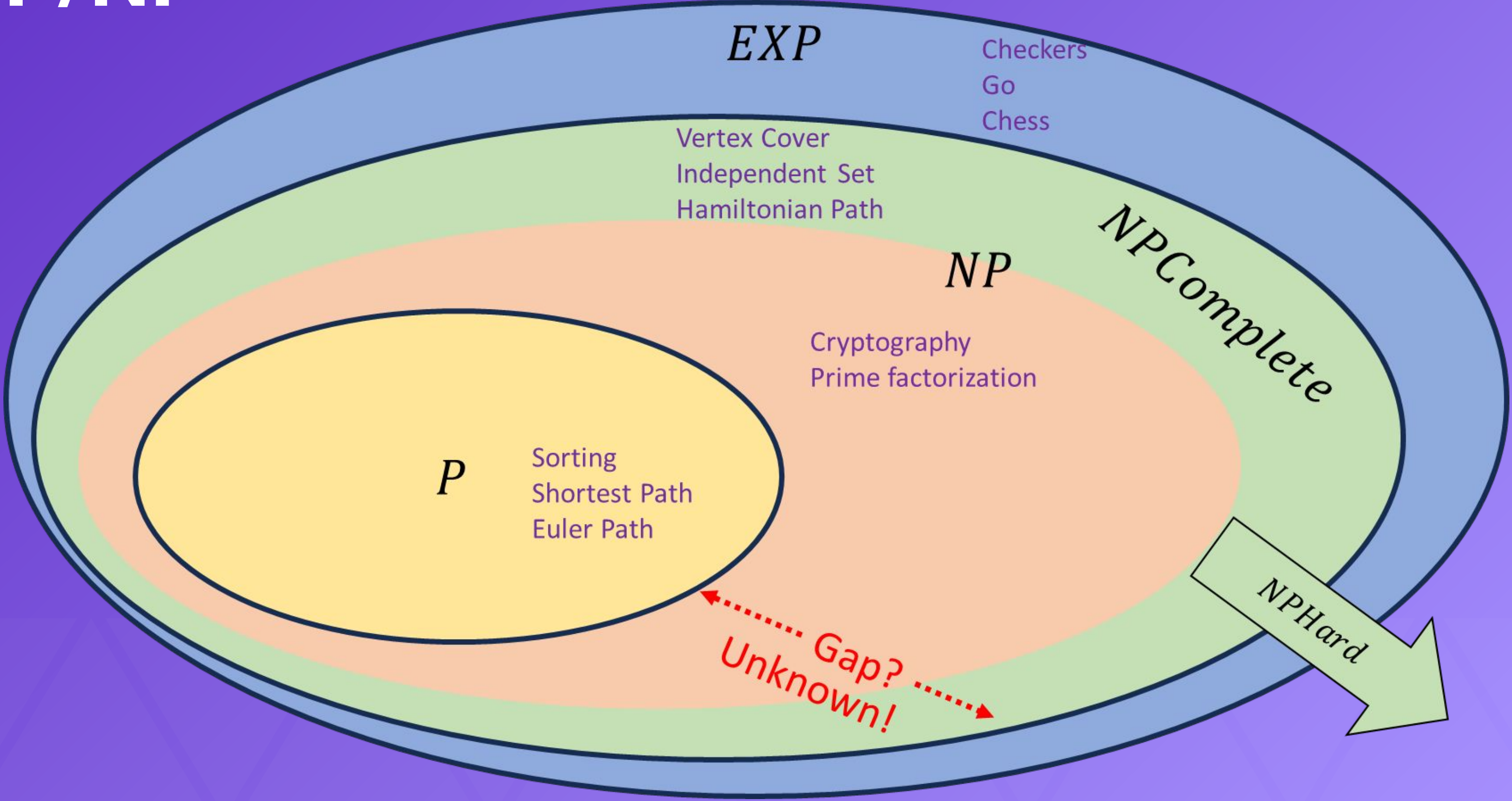
P/NP

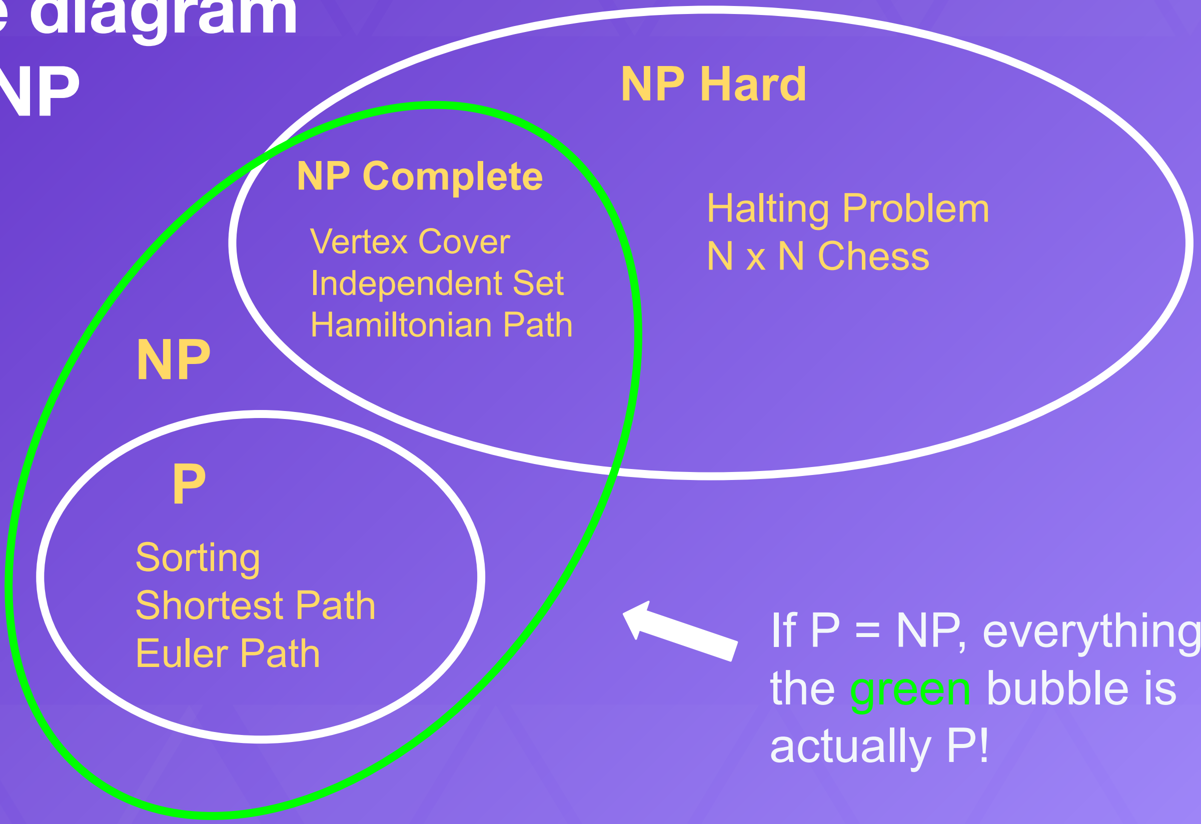
CSE 332 – Section 10



P/NP


EXP
Checkers
Go
Chess
Vertex Cover
Independent Set
Hamiltonian Path
NPComplete
NP
Cryptography
Prime factorization
P
Sorting
Shortest Path
Euler Path
Gap?
Unknown!
NPHard

An alternate diagram of P/NP



If P = NP, everything in the green bubble is actually P!

Complexity Classes (Question 0)

- Complexity Class:
    - A set of problems categorized by an asymptotic **upper bound** on the fastest algorithm that solves them
    - If a problem is solved by an algorithm that runs in O(n) time then it belongs to the complexity class for O(n) and O(n log n) and O(n^2), but not O(log n).
- What does P stand for?
- What is NP stand for?
- What is the definition of P?
- What is the definition of NP?

Complexity Classes (Question 0)

- Complexity Class:
    - A set of problems categorized by an asymptotic **upper bound** on the fastest algorithm with solves them
    - If a problem is solved by an algorithm that runs in O(n) time then it belongs to the complexity class for O(n) and O(n log n) and O(n^2), but not O(log n).
- What does P stand for?
    - Polynomial Time
- What is NP stand for?
    - Non-deterministic Polynomial Time
- What is the definition of P?
    - The set of problems which have polynomial time algorithms to solve them (running time is O(n^p))
- What is the definition of NP?
    - The set of problems which have polynomial time verifiers (can check answers in polynomial time)
    - Equivalently, the set of problems which have polynomial time non-deterministic algorithms

P & NP Membership (Question 1)

- How would we show that a given problem belongs to the class P?
- How would we show that a given problem belongs to the class NP?

P & NP Membership (Question 1)

- How would we show that a given problem belongs to the class P?
    - Show there exists a polynomial time algorithm which solves that problem.
- How would we show that a given problem belongs to the class NP?
    - Show that there exists a polynomial time algorithm which verifies that problem.
    - Note: you could also show a problem belongs to a subset of NP (e.g. P), but for today we want to practice writing verifiers!

P & NP Membership (Question 2)

- Consider the following problems:
  - Problem A: Given a list of 2-dimensional points, return true or false to indicate whether some pair of points have a distance of less than 5.
  - Problem B: Given a list of integers, return true or false to indicate whether any items are duplicated (so return true if some value appears at least twice, false if all are distinct).
  - Problem C: Given a weighted graph, a pair of nodes X and Y, and a number k, return true or false to indicate whether there is a path from X to Y with a cost of at least k
- Show that A and B belong to P
- Show that A, B, and C belong to NP
  -

Problem A is in P

- Given a list of 2-dimensional points, return true or false to indicate whether some pair of points have a distance of less than 5.

Algorithm: For each point, find its distance to every other point. If the distance is ever less than 5, return true. After doing this for all points, return false.

Running time: O(n^2)

Problem A is in NP

- Given a list of 2-dimensional points, return true or false to indicate whether some pair of points have a distance of less than 5.

Verification Algorithm: Given a candidate pair of points, verify that those points are both in the list, then verify that their distance is less than 5.

Running time: O(n)

Problem B is in P

- Given a list of integers, return true or false to indicate whether any items are duplicated (so return true if some value appears at least twice, false if all are distinct).

Algorithm: Sort the list, then check if any two neighboring items match.

Running time: O(n log n)

Problem B is in NP

- Given a list of integers, return true or false to indicate whether any items are duplicated (so return true if some value appears at least twice, false if all are distinct).

Verification Algorithm: Given a candidate value, check that the value appears at least twice in the list.

Running time: O(n)

Problem C is in NP

- Given a weighted graph, a pair of nodes X and Y, and a number k, return true or false to indicate whether there is a path from X to Y with a cost of at least k

Verification Algorithm: Given a candidate path, check that it is a valid path (all adjacent nodes in the path form edges in the graph), and check that the sum of those edge weights is at least the value k.

Running time: O(V + E)

NP-Hard, NP-Complete (Question 3)

- What is the definition of NP-Hard?
    -
- What is the definition of NP-Complete?
    -

NP-Hard, NP-Complete

- What is the definition of NP-Hard?
    - The problems that are “at least as hard” as any in NP
    - The set of problems for which there exists a polynomial time reduction from any NP problem to it
- What is the definition of NP-Complete?
    - The intersection of NP and NP-Hard

NP-Hard, NP-Complete (Question 4)

- How do you show that a problem belongs to NP-Hard
    -
- How do you show that a problem belongs to NP-Complete
    -

NP-Hard, NP-Complete

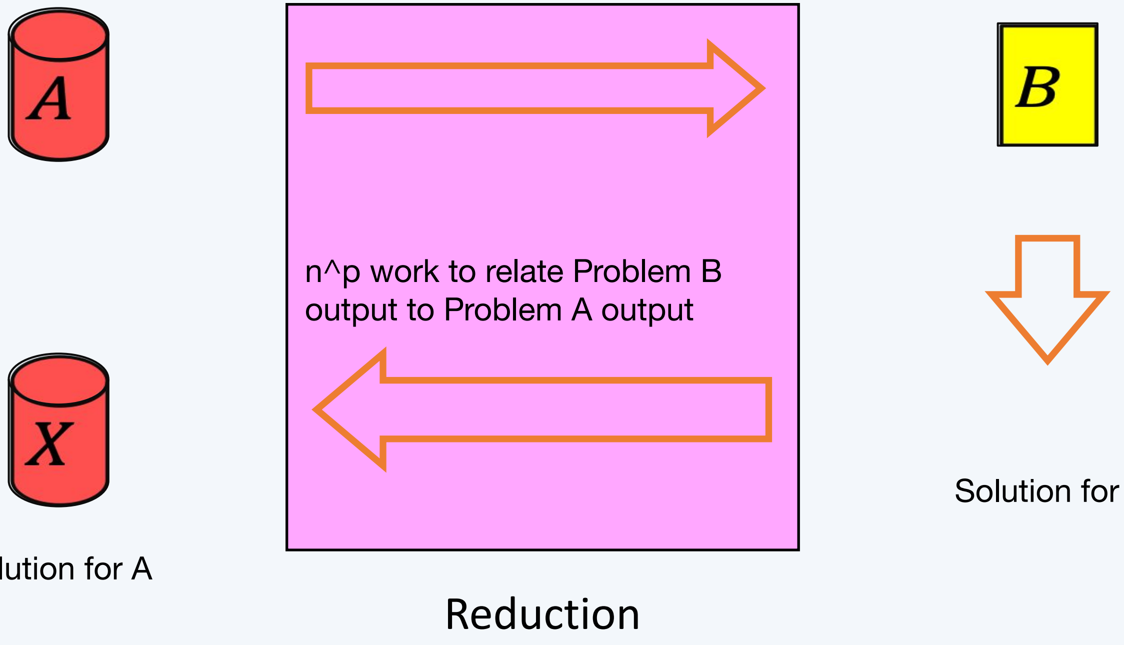
- How do you show that a problem belongs to NP-Hard
  - Reduce a preexisting NP-Hard problem to it
- How do you show that a problem belongs to NP-Complete
  - Show that it belongs to NP and NP-Hard
  - Find a polynomial time verification algorithm and reduce preexisting NP-Hard problem to it

Reduction

- A reduction from problem A to problem B is a way of solving A using B as a subroutine
  - Plugging in any algorithm for B finishes an algorithm for A
- A polynomial time reduction is a reduction which requires only a polynomial amount of additional work (ignoring the running time of the algorithm for B, the running time is polynomial)
  - Plugging in a polynomial time algorithm for B finishes a polynomial time algorithm for A
- Most Importantly:
  - If there is a polynomial time reduction from A to B, then if B has a polynomial time solution, then A does as well.
  - *If A is NP-Hard, since every NP problem reduces to it in polynomial time, reducing A to B means solving B in polynomial time allows us to solve EVERY NP problem in polynomial time*

Reductions Visual

“A polynomial time reduces to B”



n^p work to relate Problem A input to Problem B input
A
B
n^p work to relate Problem B output to Problem A output
X
Solution for A
Solution for B
Reduction

Reductions and NP-Hard

- Given there is a polynomial time reduction from A to B, then if B has a polynomial time solution, then A does as well.
- If A is NP-Hard, since every NP problem reduces to it in polynomial time, a polynomial time algorithm for solving A finishes polynomial time algorithms for ALL of NP.
- Because of the reduction, a polynomial time algorithm for B is the only missing piece of a polynomial time algorithm for A
    - So if B has a polynomial time algorithm, then so does A, and therefore all of NP does as well.

NP-Complete

- If some problem A is NP-Complete then:
    - Because A belongs to NP, if there does not exist a polynomial time algorithm for A then we can conclude P does not equal NP
    - Because A belongs to NP hard, if there does exist a polynomial time algorithm for A then we can solve EVERY NP problem in polynomial time, and so P = NP.
    - Answering the question “can A be solved in polynomial time?” gives us the answer for the entirety of P=NP

Practice - True/False

- If A polynomial-time reduces to B and B is NP-Hard then A is NP-Hard.
- If B is NP-Hard and there exists a polynomial time algorithm for B, then P=NP
- If B is NP-Hard and there does not exist a polynomial time algorithm for B, then P does not equal NP
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is NP-Hard, then C is NP-Hard.
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is NP-Complete, then C is NP-Complete.
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is in EXP, then C is EXP.
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is in P, then C is P.
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is in NP, and C is in P, then P=NP
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is in NP-Hard, and C is in P, then P=NP

Practice - True/False

- If A polynomial-time reduces to B and B is NP-Hard then A is NP-Hard. False
- If B is NP-Hard and there exists a polynomial time algorithm for B, then P=NP. True
- If B is NP-Hard and there does not exist a polynomial time algorithm for B, then P does not equal NP. False
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is NP-Hard, then C is NP-Hard. True
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is NP-Complete, then C is NP-Complete. False
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is in EXP, then C is EXP. False
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is in P, then C is P. False
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is in NP, and C is in P, then P=NP. False
- If A reduces to B in polynomial time, and B reduces to C in polynomial time, and A is in NP-Hard, and C is in P, then P=NP. True

Thank You!