Presentations

- Thursday 4/22, Tuesday 4/27, and Tuesday 5/4 @ 8am
- 6-7 minutes
- Summarize only the key critical concepts
  - The overview of the topic/problem may be more important than the details of your solution/implementation
- Use figures and images, keep text minimal

- Demo software if time permits

- Be ready to present on Thursday with slides

- Fully remote class

Paper and Software

- Due Tuesday 4/27

P vs NP

- P is the set of problems that can be solved in polynomial time
- NP is the set of problems whose solutions can be verified in polynomial time
    - Includes P
    - We think there are problems in NP that are not in P, but we don't know for sure

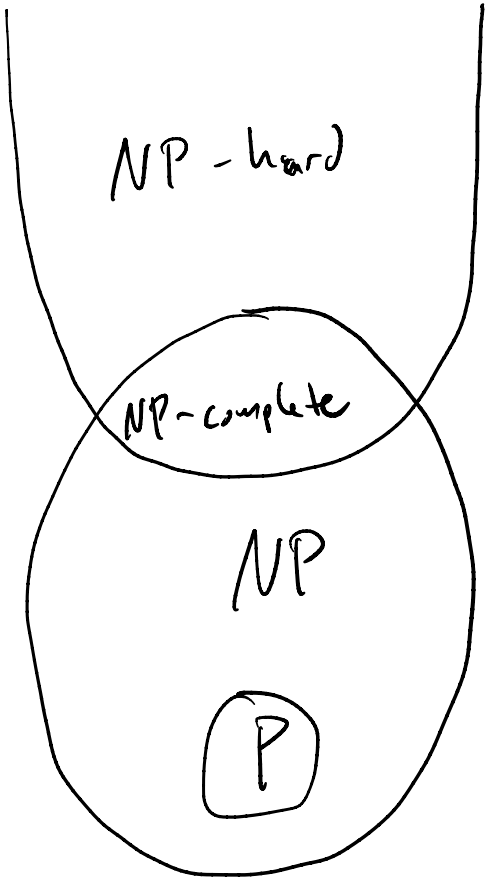
- NP-complete - problems in NP, and every other problem in NP can be reduced to these problems in polynomial time

- How would we prove $P = NP$?
  - Prove that some NP-complete problem can be solved in polynomial time

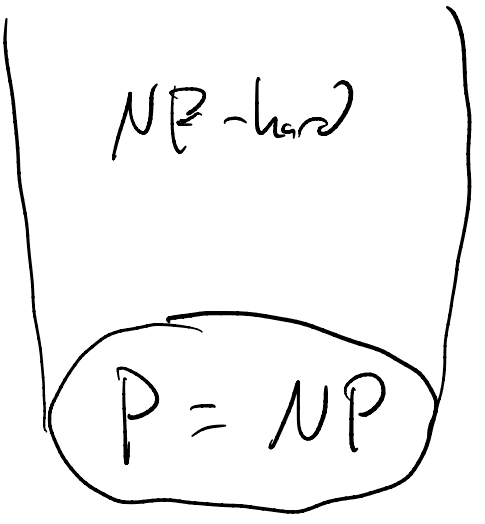
- How would we prove $P \neq NP$?
  - Prove it is impossible to solve some NP-complete problem in polynomial time

- NP-hard
  - Problems that are at least as hard as NP
  - Every problem in NP can be reduced to a problem in NP-hard
  - NP-hard problems do not have to be verifiable in polynomial time

- NP-hard includes optimization problems
    - There is significant research into NP-hard approximation algorithms where solutions are "good enough"



VS.



Patterns, Strategies, and Other Takeaways

- Nested loops often indicate $O(n^2)$
- Divide and conquer is often $O(n \log n)$ or $O(\log n)$
- Binary trees often get us $O(\log n)$ behavior
- Graphs involve $|V|$ and $|E|$, so sparsity matters
  - Is $|E|$ close to $|V|^2$

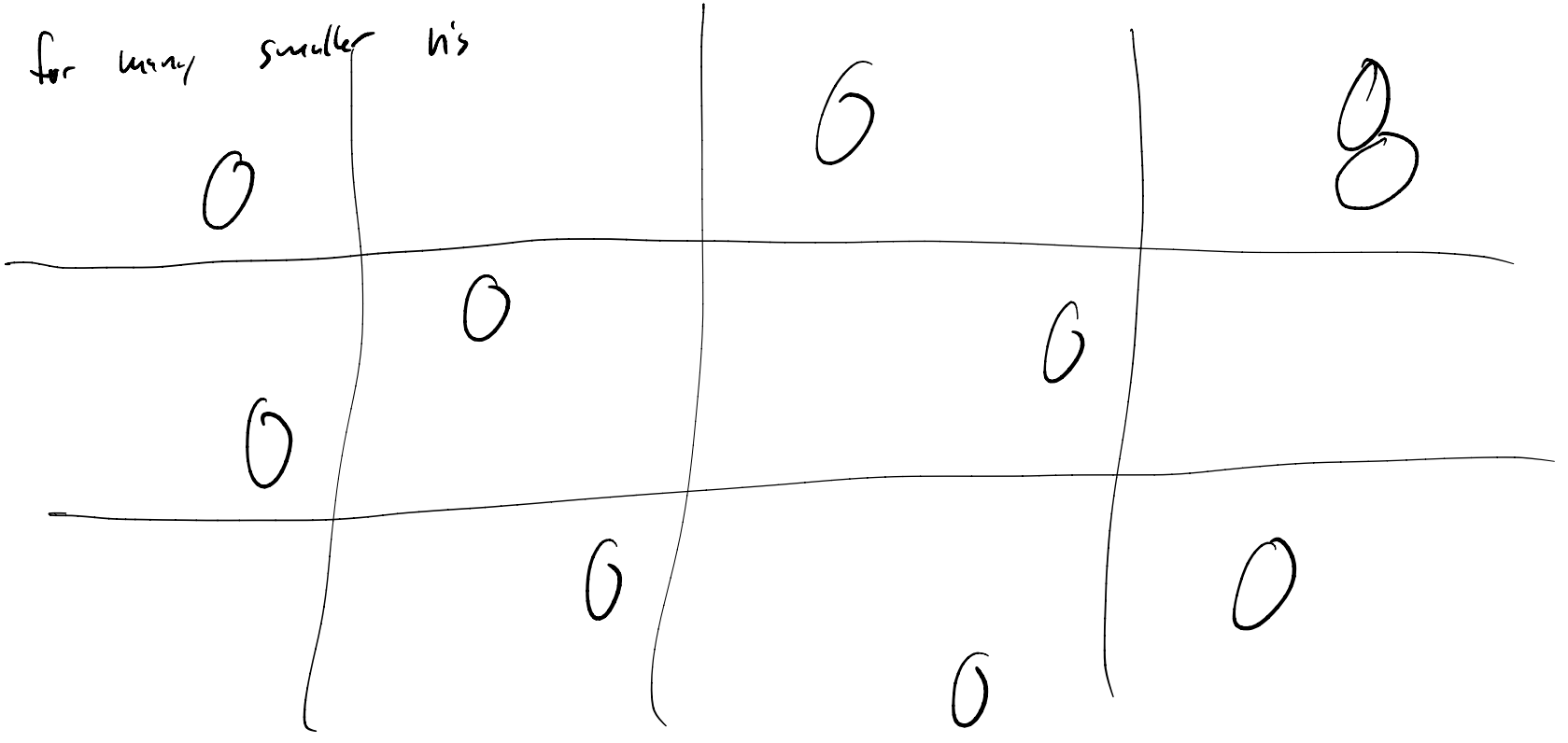
- Dynamic programming can cut down on re-calculating solutions over and over

- Greedy algorithms
  - Sort first or put things in a priority queue, then work with the best things first

Why Study that Stuff?

- Recognize impractical solutions before coding them up
- Understanding documentation
  - Libraries that provide data structures often mention the time complexity of operations
- Understanding CS papers
- Writing CS papers

Collision Detection is $O(n^2)$ but spacial partitioning can make for many smaller n's



Ordering of Common Time Complexities

- $O(1)$
- $O(\log n)$
- $O(n)$
- $O(n \log n)$
- $O(n^2)$, $O(n^3)$, ... $O(n^k)$
- $O(2^n)$
- $O(n!)$