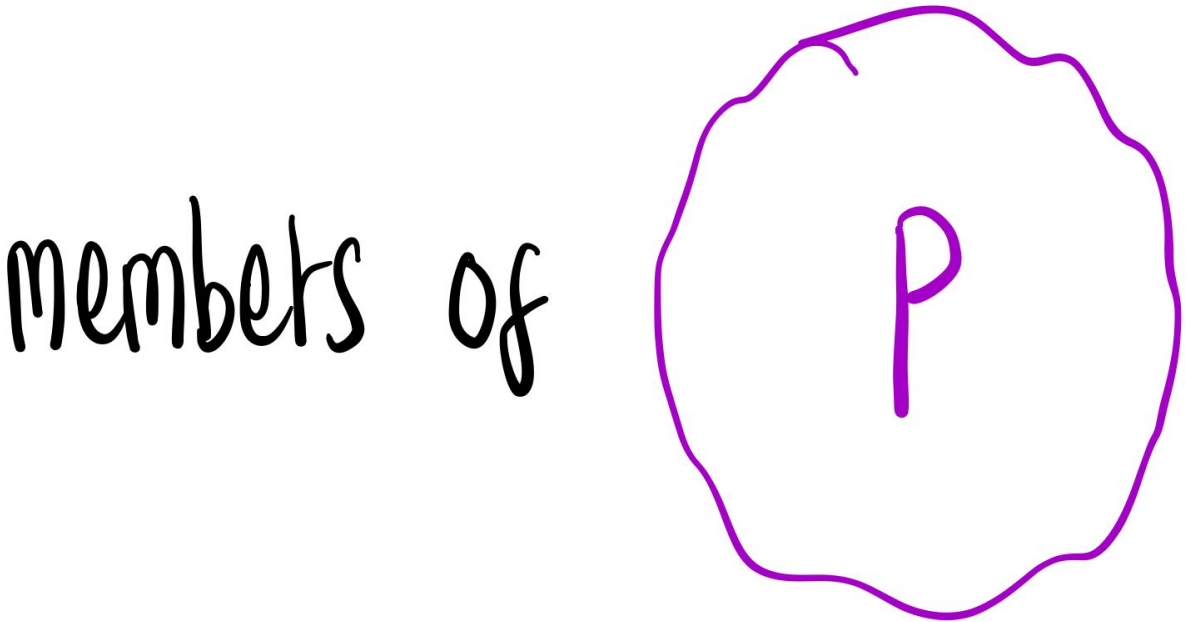
P v.s. NP and Approximation Algorithms

August 14th, 2021 (Class #6)

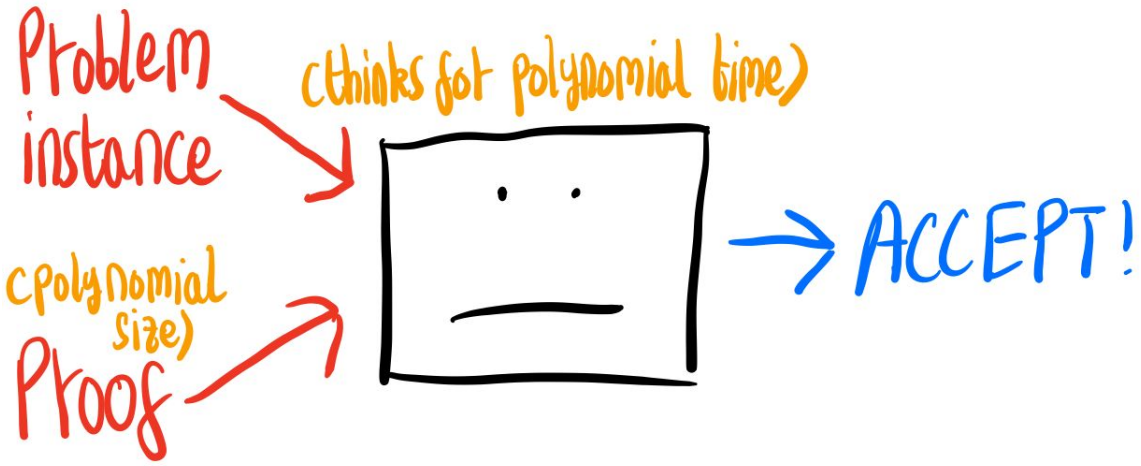
Our Best Results

Problem	Running time
Multiplication	$O(n^{\log_2 3})$
Sorting	$O(n \log n)$
Unweighted SP	$O(n+m)$
Max Disj. Intervals	$O(n)$
Weighted SP	$O(m \log n)$

We've shown that all these problems are...



What is NP?



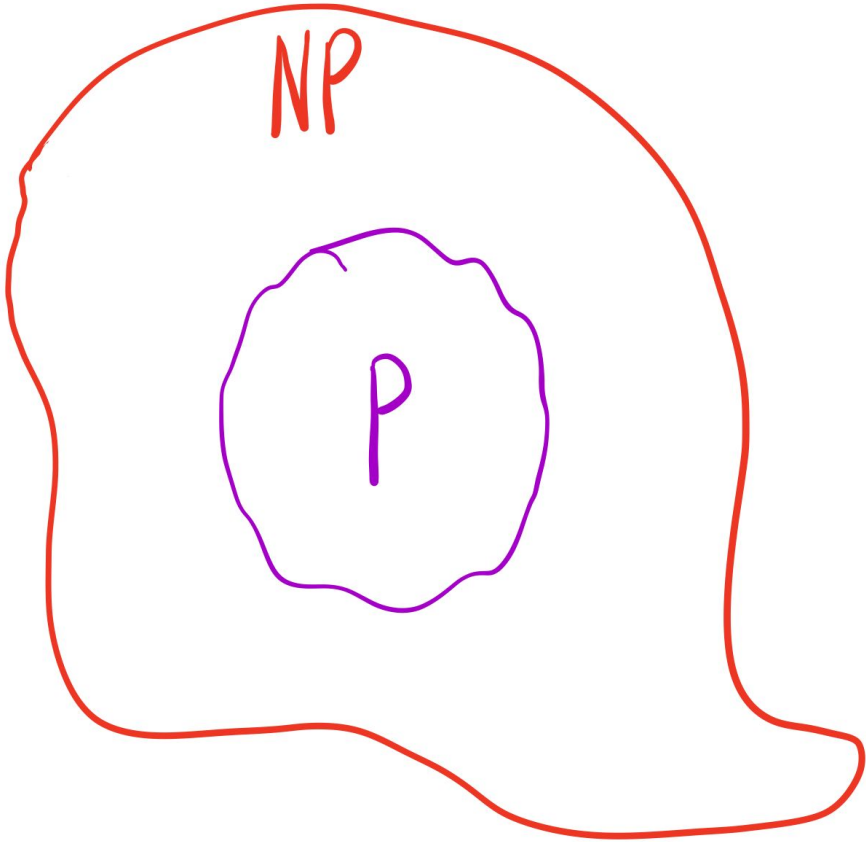
If the true answer is YES, then  will ACCEPT for some proof.

Otherwise, always REJECT.

The P v.s. NP Question

NP: If we can verify an answer quickly, (check a proof)

P: Can we find it quickly? (come up with a proof)

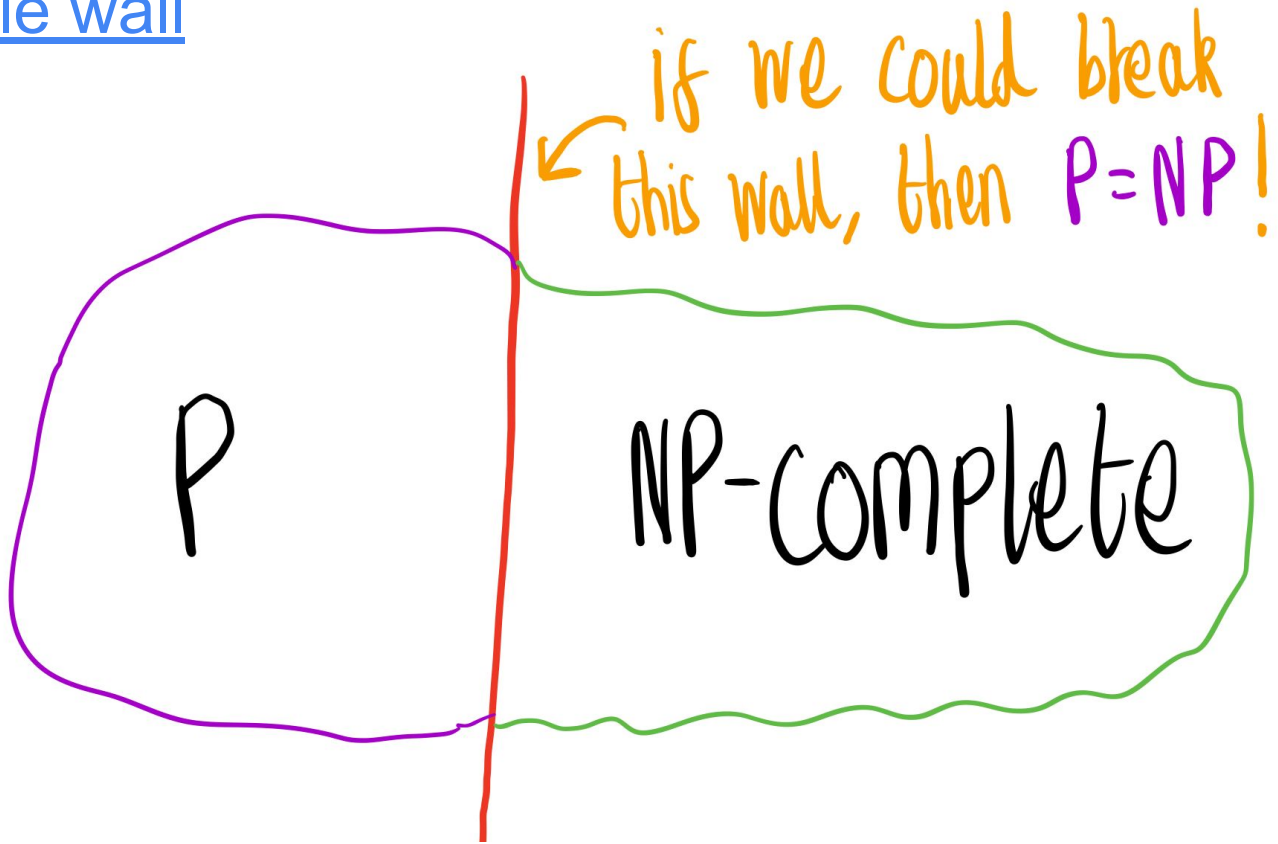


NP-complete problems

If we could (quickly) solve an NP-Complete problem, we could (quickly) solve every problem in NP.

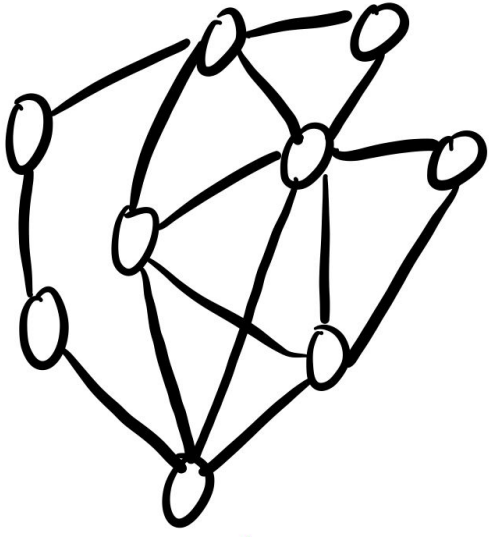
The surprise: There exist very reasonable sounding problems that are provably NP-Complete.

The invisible wall



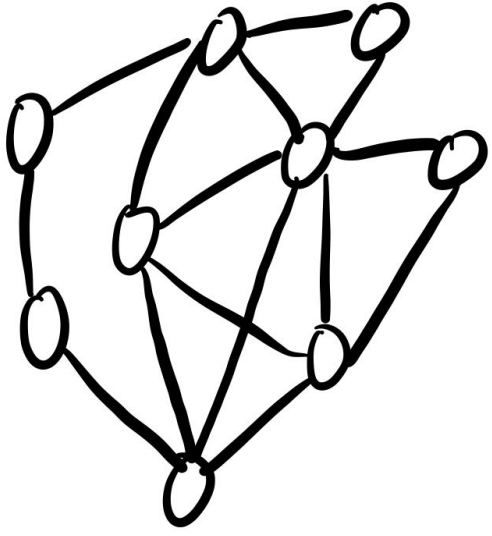
Eulerian Path v.s. Hamiltonian Path

polynomial time



Use every edge once.

exponential time



Visit every node once.

2-SAT v.s. 3-SAT

polynomial time

(A or B) and

(A or not C) and

(not B or not A) and

(not B or C)

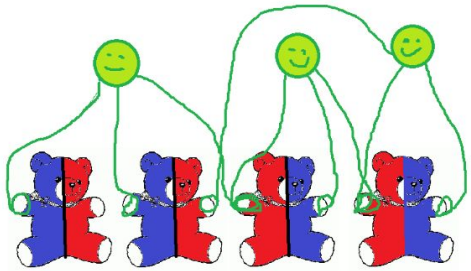
2 variables per clause

exponential time

(A or B or C) and

(C or not B or not D) and

...



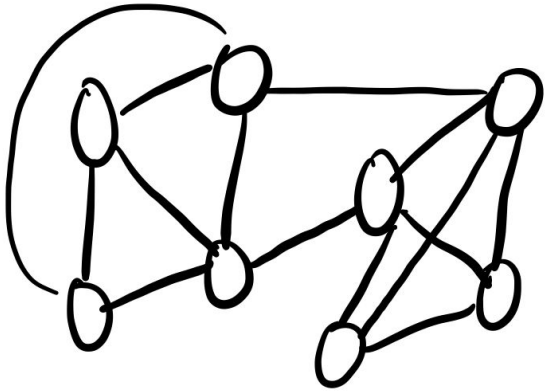
3 variables per clause

Credit: David M, https://math.stackexchange.com/questions/86210/what-is-the-3-sat-problem

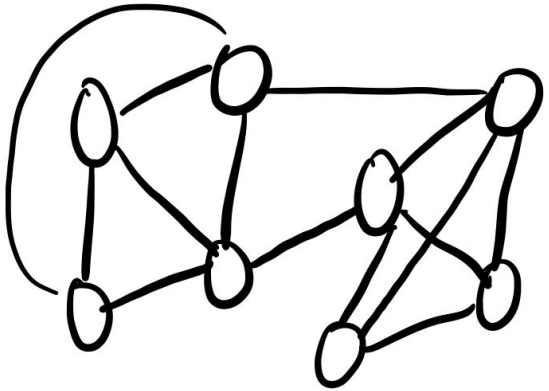
Min Cut v.s. Max Cut

polynomial time

exponential time

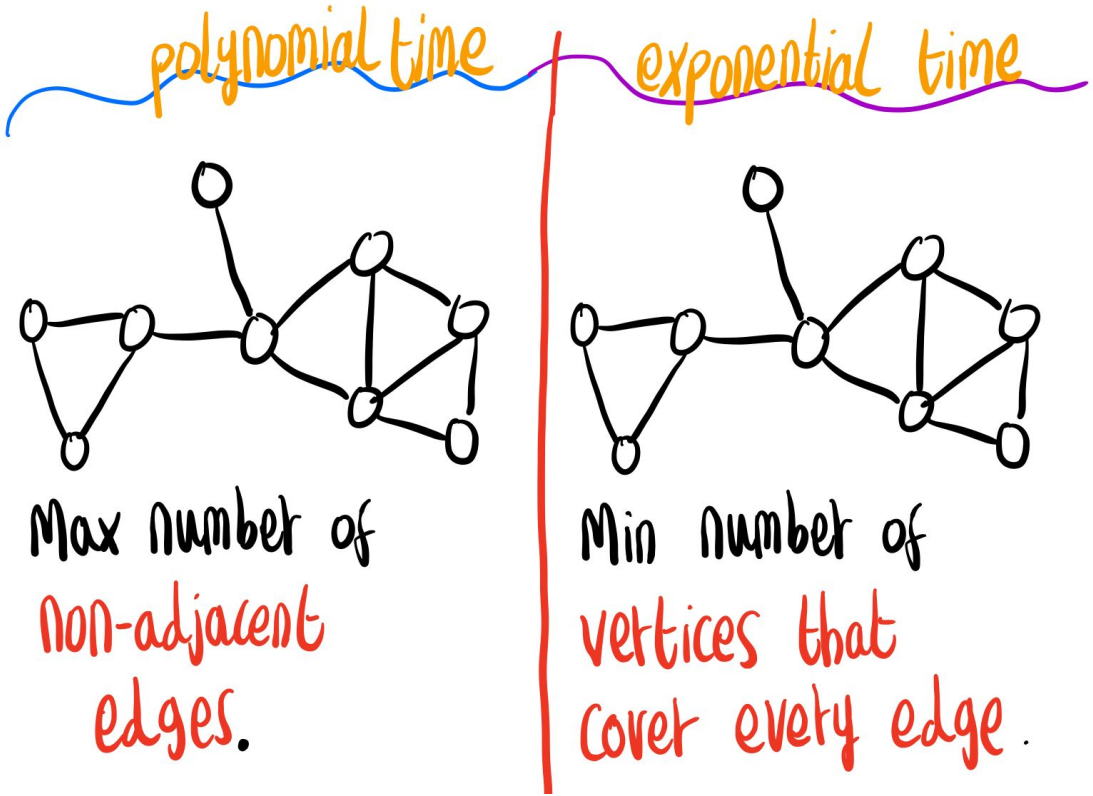


Cut with fewest crossing edges.



Cut with most crossing edges.

Maximum Matching v.s. Minimum Vertex Cover



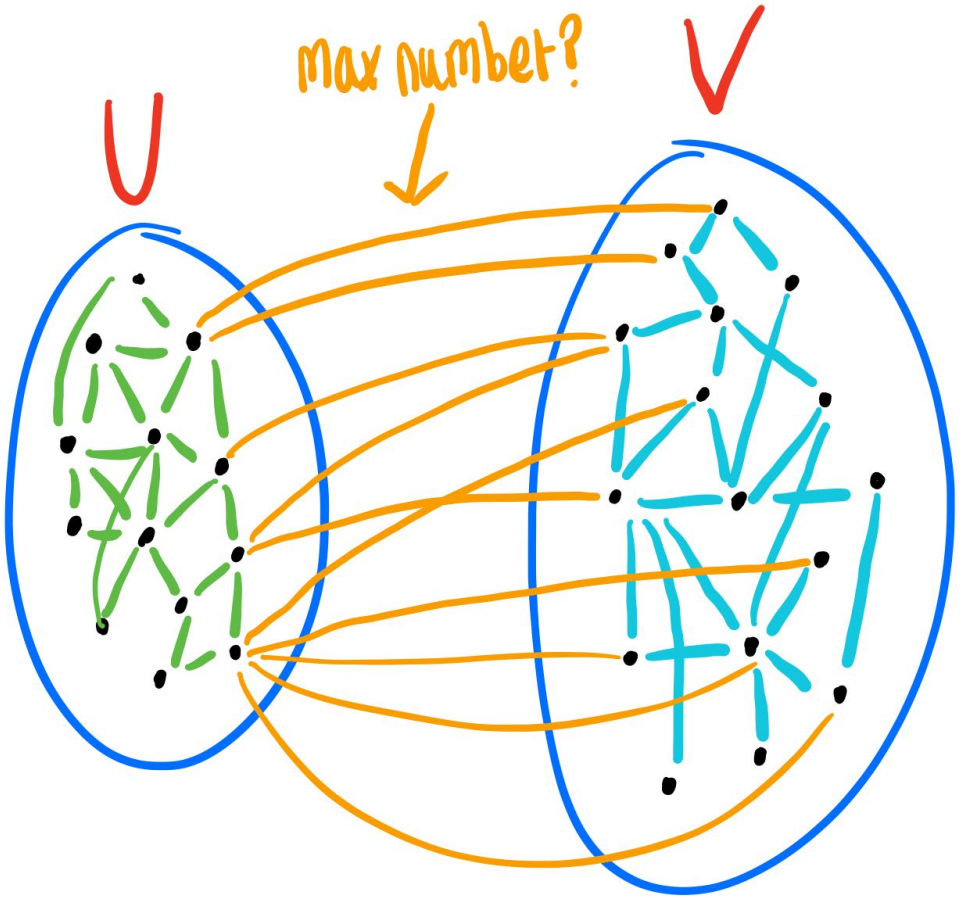
Break for 5 minutes

Approximation Algorithms

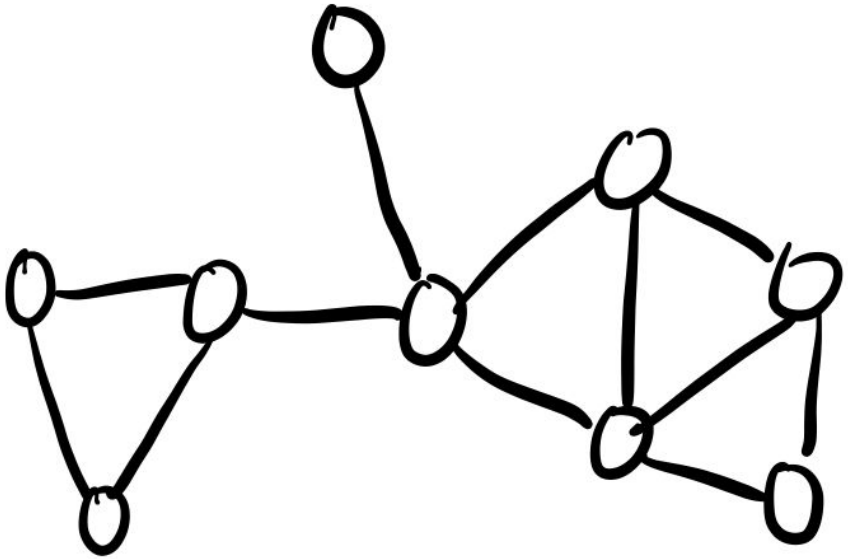
For "optimisation problems":

Get within a multiplicative factor of the optimal.

Approximation Algorithm for Max Cut?



Approximation Algorithm for Vertex Cover?



Min number of vertices that cover every edge.

The big ideas of algorithm design

- Correctness

  Correct answers to well-defined problems.

- Asymptotic complexity (measure of efficiency)

  How does number of steps SCALE with input size?

- Worst-case analysis

  Want to do well on every possible input!

→ Leads to an incredibly rich design space...

as we've seen!

Further Exploration

Exploring more algorithms...

MIT OCW (Profs. Devadas & Demaine)

Coursera (Prof. Roughgarden)

Further Exploration

Implementing algorithms!

- Just use your favorite language and play around.
  - * Python
- Competitive Programming
  - * Codeforces, Atcoder, etc.
  - * CP Handbook by Laaksonen
  - * USACO

Hope you've enjoyed the classes!