CS146: Data Structures and Algorithms Lecture 18

NP-COMPLETENESS

INSTRUCTOR: KATERINA POTIKA
CS SJSU

Algorithm Design Patterns and Anti-Patterns

- Algorithm design patterns. Ex.

Greed.	$O(n \log n)$ interval scheduling.
Divide-and-conquer.	$O(n \log n)$ Mergesort.
Dynamic programming.	$O(n W)$ 0-1 Knapsack.
Randomization.	$O(n\lg n)$ Quicksort

- Algorithm design anti-patterns.

NP-completeness.	$O(n^k)$ algorithm unlikely.
PSPACE-completeness.	$O(n^k)$ certification algorithm unlikely.
Undecidability.	No algorithm possible.

NP-Completeness (Ch 34 & Ch8 of DPV)

- Some problems are *intractable*:
  as they grow large, we are unable to solve them in reasonable time
- What constitutes reasonable time? Standard working definition: *polynomial time*
  - On an input of size n the worst-case running time is $O(n^k)$ for some constant k
  - Polynomial time: $O(n^2)$, $O(n^3)$, $O(1)$, $O(n \lg n)$
  - Not in polynomial time: $O(2^n)$, $O(n^n)$, $O(n!)$

Classify Problems

- Desiderata. Classify problems according to those that can be solved in polynomial-time and those that cannot.

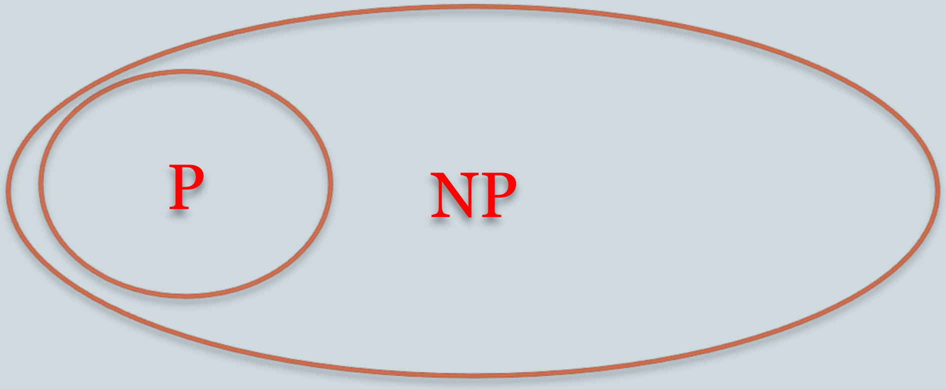
- Provably requires exponential-time.
  - Given a Turing machine, does it halt in at most k steps?
  - Given a board position in an n-by-n generalization of chess, can black guarantee a win?

- Frustrating news. Huge number of fundamental problems have defied classification for decades.

- Show that these fundamental problems are "computationally equivalent" and appear to be different manifestations of one really hard problem.

P and NP

- **P** is set of problems that can be solved in polynomial time
- **NP** (*nondeterministic polynomial time*) is the set of problems that can be solved in polynomial time by a *nondeterministic* computer
  - *What is that? Next..*



Non-determinism

- Think of a non-deterministic computer as a computer that magically “guesses” a solution, then has to verify that it is correct
  - If a solution exists, computer always guesses it
  - One way to imagine it: a parallel computer that can freely spawn an infinite number of processes
    - Have one processor work on each possible solution
    - All processors attempt to verify that their solution works
    - If a processor finds it has a working solution
  - So: **NP** = problems *verifiable* in polynomial time

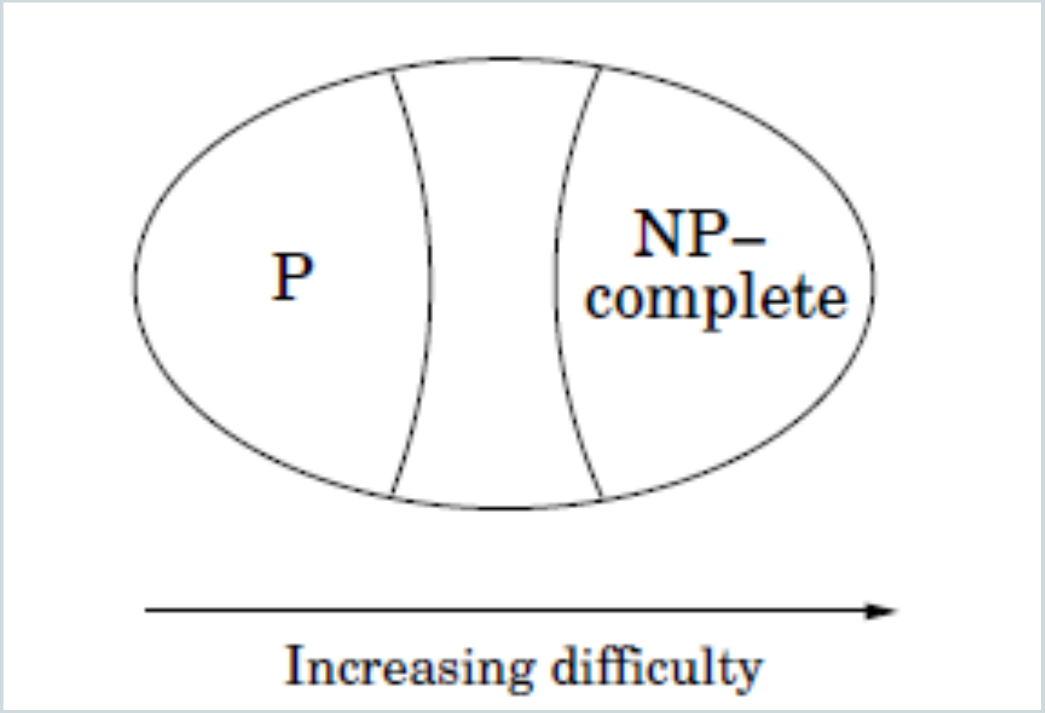
P and NP

- Summary so far:
  - **P** = problems that can be **solved** in polynomial time
  - **NP** = problems for which a solution can be **verified** in polynomial time
  - Unknown whether **P = NP** (most suspect not)
- Hamiltonian-cycle problem is in **NP**:
  - Cannot solve in polynomial time
  - Easy to verify solution in polynomial time (*How?*)

NP-Complete Problems

- We will see that NP-Complete problems are the "hardest" problems in NP:
  - If any *one* NP-Complete problem can be solved in polynomial time…
  - …then *every* NP-Complete problem can be solved in polynomial time…
  - …and in fact *every* problem in **NP** can be solved in polynomial time (which would show **P** = **NP**)
  - Thus: solve hamiltonian-cycle in O(n^{100}) time, you've proved that **P** = **NP**. Retire rich & famous (at least 1 Million dollars…).

Assuming P≠NP

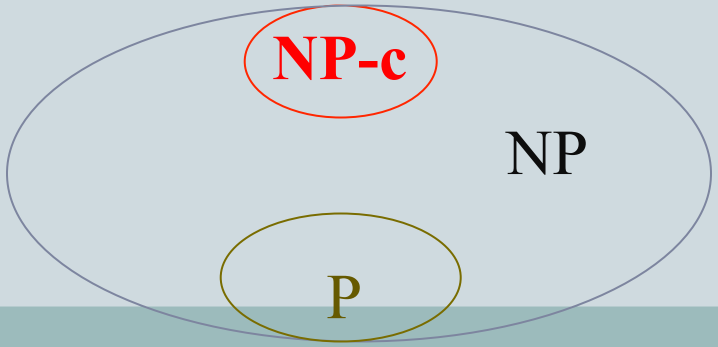


NP-complete Definition

- If A is *polynomial-time reducible* to B, we denote this $A \leq_p B$
- Definition: B is NP-Complete (also steps for proofs)
  1. $B \in$ **NP** and
  2. $A \leq_p B,\ \forall A \in$ **NP** (all problems A are reducible to B)
  - If we don't have $B \in$ **NP**, then B NP-hard
- If $A \leq_p B$ and A is NP-Complete, B is also NP-Complete
  - This is the *key idea* !!!

First NP-complete

- The technique relies on having a known NP-complete problem
- SAT: the first known NP-complete problem, as proved by Stephen Cook in 1971.
- Class **of NP-complete** problems, are the hardest in NP.



Certifiers and Certificates: 3-Satisfiability

SAT. Given a CNF formula Φ, is there a satisfying assignment?

Certificate. An assignment of truth values to the n boolean variables.

- Certifier. Check that each clause in Φ has at least one true literal.

- Ex.

$$\left(\overline{x_1} \lor x_2 \lor x_3\right) \land \left(x_1 \lor \overline{x_2} \lor x_3\right) \land \left(x_1 \lor x_2 \lor x_4\right) \land \left(\overline{x_1} \lor \overline{x_3} \lor \overline{x_4}\right)$$

instance s

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

certificate t

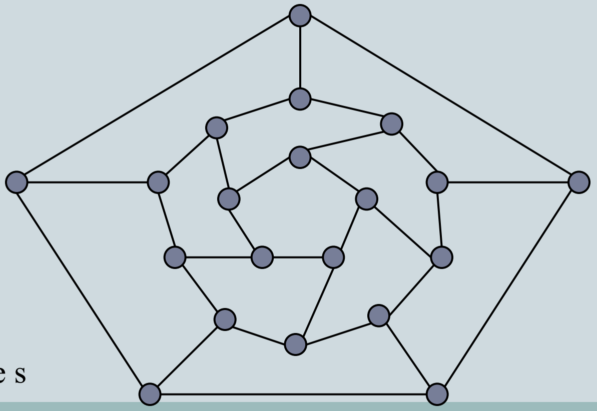
- Conclusion. SAT is in NP.

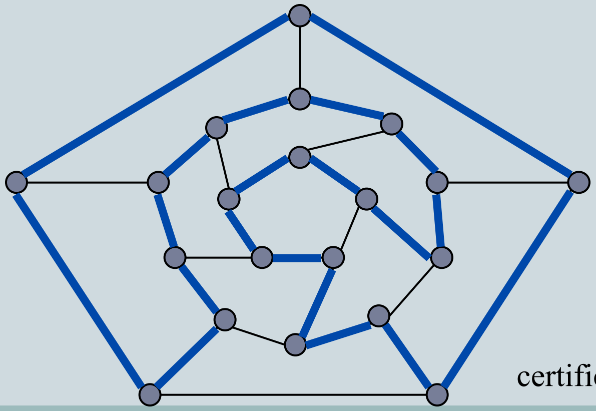
Certifiers and Certificates: Hamiltonian Cycle

HAM-CYCLE. Given an undirected graph G = (V, E), does there exist a simple cycle C that visits every node?

Certificate. A permutation of the n nodes.

- Certifier. Check that the permutation contains each node in V exactly once, and that there is an edge between each pair of adjacent nodes in the permutation.
- Conclusion. HAM-CYCLE is in NP.


instance s


certificate t

P, NP, EXP

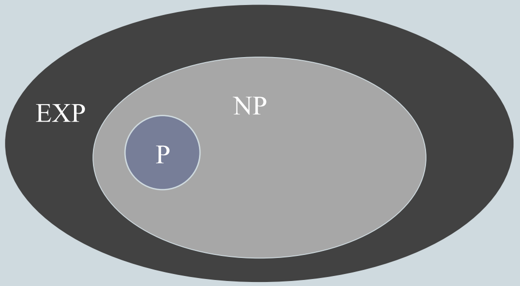
- P. Decision problems for which there is a poly-time algorithm.
- EXP. Decision problems for which there is an exponential-time algorithm.
- NP. Decision problems for which there is a poly-time certifier.

- Claim. P $\subseteq$ NP.
- Pf. Consider any problem X in P.
  - By definition, there exists a poly-time algorithm A(s) that solves X.
  - Certificate: t = ε, certifier C(s, t) = A(s). ▪

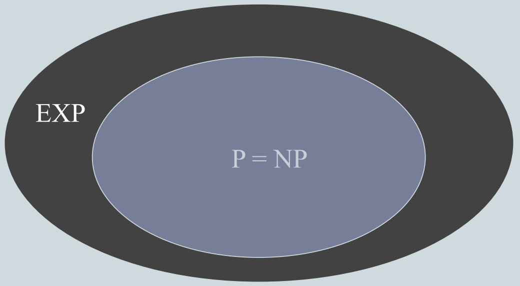
- Claim. NP $\subseteq$ EXP.
- Pf. Consider any problem X in NP.
  - By definition, there exists a poly-time certifier C(s, t) for X.
  - To solve input s, run C(s, t) on all strings t with $|t| \leq p(|s|)$.
  - Return `yes`, if C(s, t) returns `yes` for any of these. ▪

The Main Question: P Versus NP

- Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
  - Is the decision problem as easy as the certification problem?
  - Clay $1 million prize.



If P NP



If P = NP

would break RSA cryptography
(and potentially collapse economy)

↙

- If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, …
- If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, …

- Consensus opinion on P = NP? Probably no.

NPC: Traveling Salesperson Problem

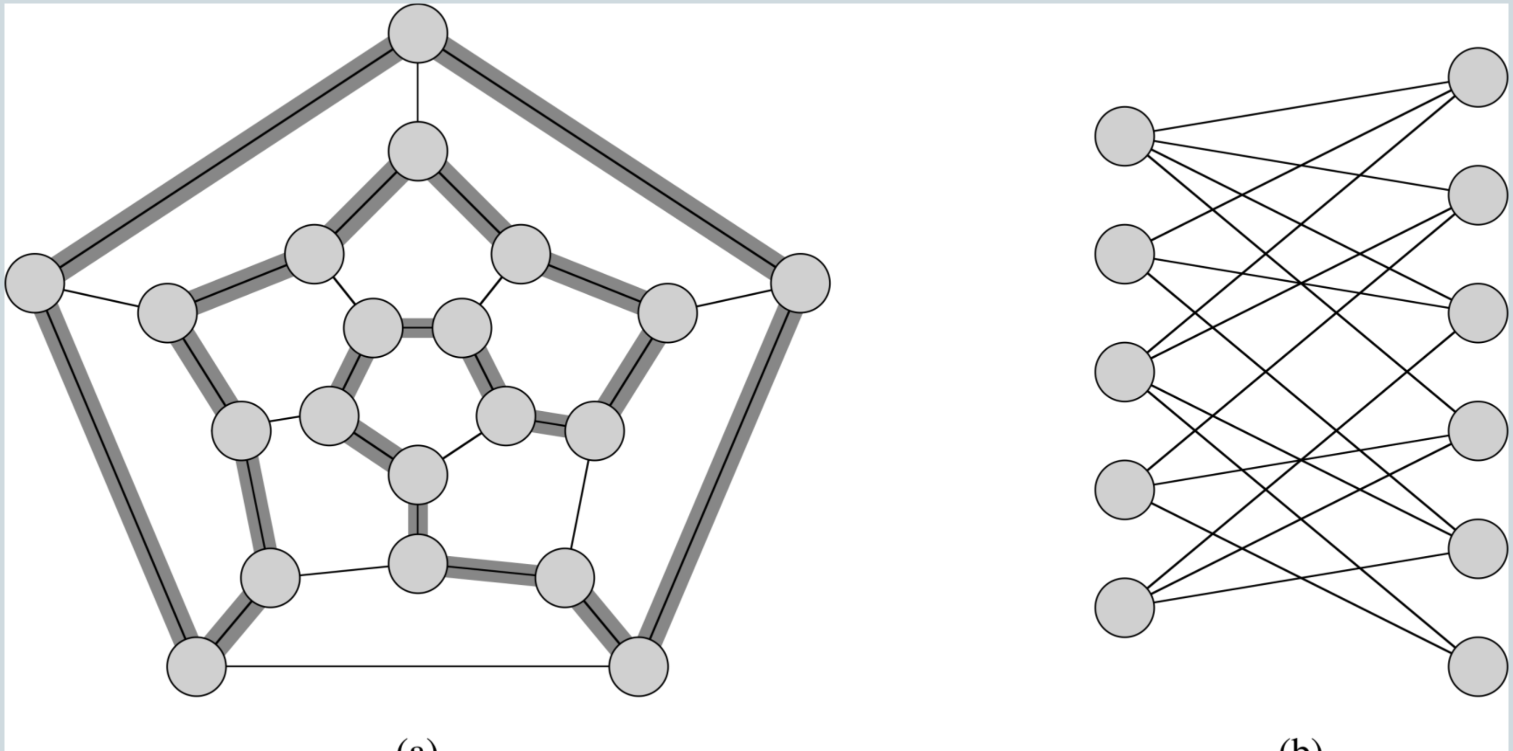
- TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?

NPC: Hamiltonian Cycles

- A *hamiltonian cycle* of an undirected graph is a simple cycle that contains every vertex
- The hamiltonian-cycle problem: given a graph G, does it have a hamiltonian cycle?
    - Draw on board: dodecahedron, odd bipartite graph (what?)
- *Describe a naïve algorithm for solving the hamiltonian-cycle problem. Running time?*

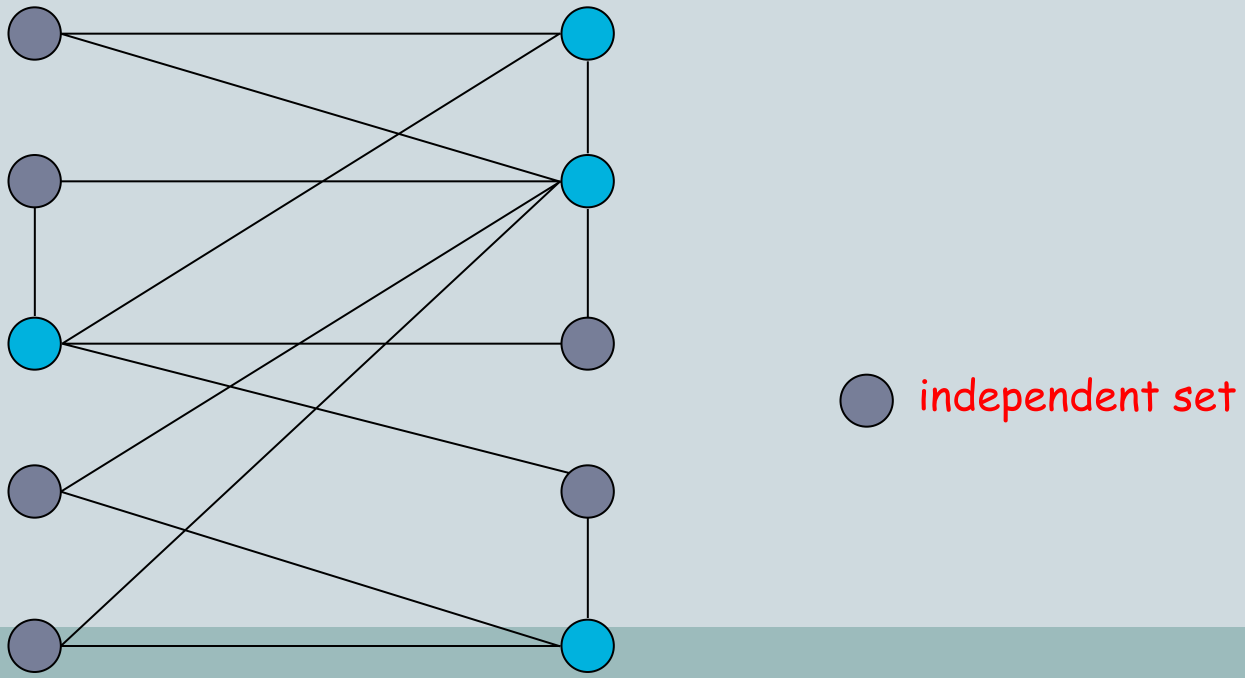
Hamiltonian cycles (example Ch34)

- a) Hamiltonian & b) non hamiltonian



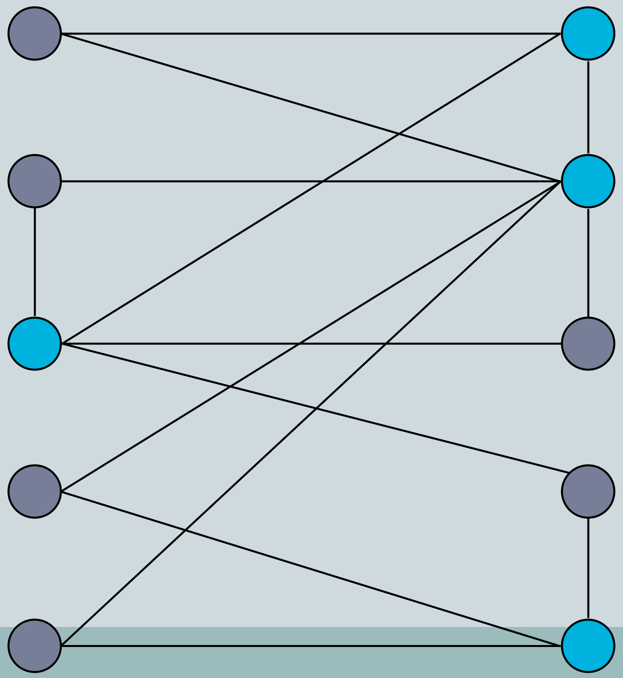
Independent Set

- INDEPENDENT SET: Given a graph G = (V, E) and an integer k, is there a subset of vertices S ⊆ V such that |S| ≥ k, and for each edge at most one of its endpoints is in S?
- Ex. Is there an independent set of size ≥ 6? Yes.
- Ex. Is there an independent set of size ≥ 7? No.



Vertex Cover

- VERTEX COVER: Given a graph G = (V, E) and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in S?
- Ex. Is there a vertex cover of size ≤ 4? Yes.
- Ex. Is there a vertex cover of size ≤ 3? No.



● vertex cover

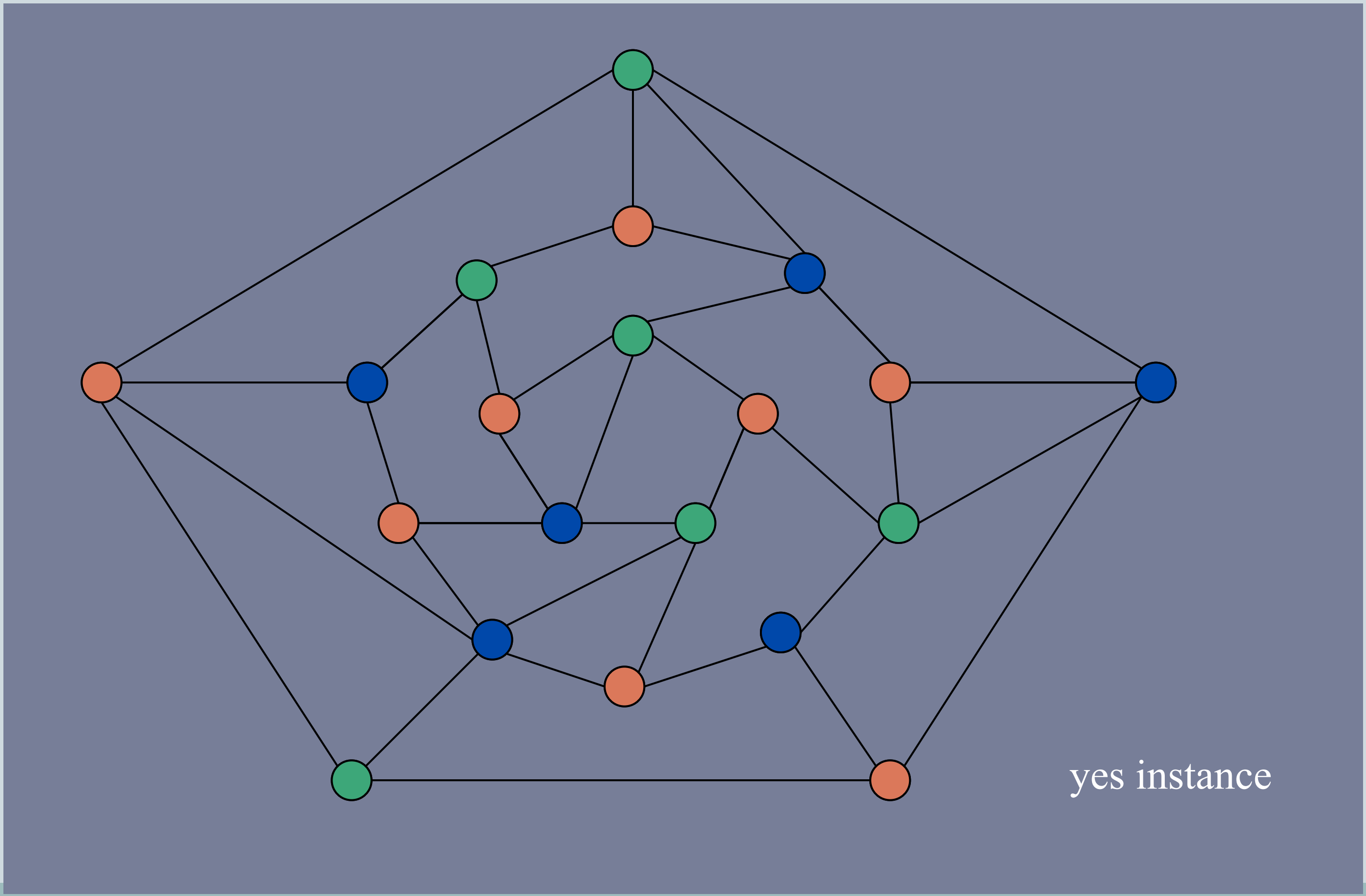
3-Dimensional Matching

- 3D-MATCHING. Given n instructors, n courses, and n times, and a list of the possible courses and times each instructor is willing to teach, is it possible to make an assignment so that all courses are taught at different times?

Instructor	Course	Time
Wayne	COS 423	MW 11-12:20
Wayne	COS 423	TTh 11-12:20
Wayne	COS 226	TTh 11-12:20
Wayne	COS 126	TTh 11-12:20
Tardos	COS 523	TTh 3-4:20
Tardos	COS 423	TTh 11-12:20
Tardos	COS 423	TTh 3-4:20
Kleinberg	COS 226	TTh 3-4:20
Kleinberg	COS 226	MW 11-12:20
Kleinberg	COS 423	MW 11-12:20

3-Colorability

- 3-COLOR: Given an undirected graph G does there exists a way to color the nodes red, green, and blue so that no adjacent nodes have the same color?



Subset Sum

- SUBSET-SUM. Given natural numbers $w_1, \ldots, w_n$ and an integer W, is there a subset that adds up to exactly W?

- Ex: { 1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344 }, W = 3754.
- Yes. 1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754.

- Remark. With arithmetic problems, input integers are encoded in binary. Polynomial reduction must be polynomial in binary encoding.

Partition

- PARTITION. Given natural numbers $v_1, \ldots, v_m$, can they be partitioned into two subsets that add up to the same value?

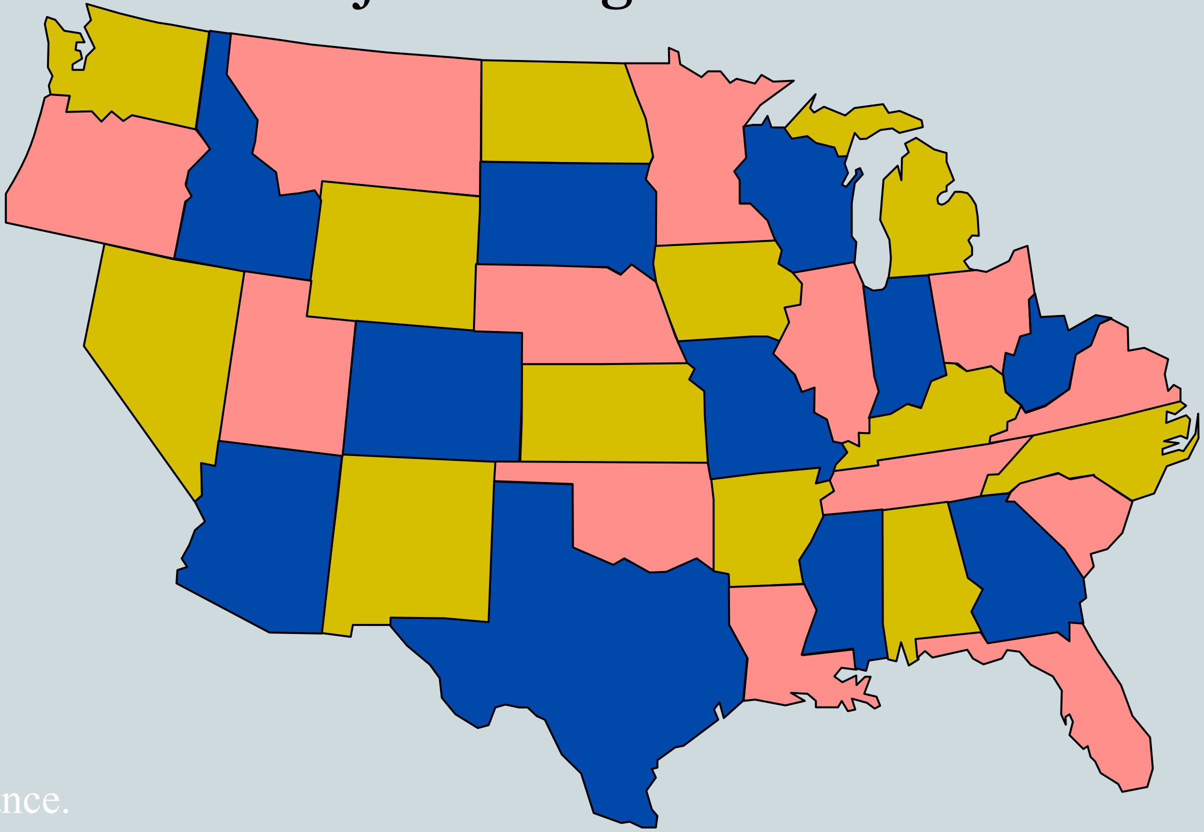
$$\tfrac{1}{2} \Sigma_i \, v_i$$

Longest simple paths

- We know that *shortest* paths in *P*
- Finding a *longest* simple path between two vertices is NP-complete.
- Determining whether a graph contains a simple path with at least a given number of edges is NP-complete.

Planar 3-Colorability

- PLANAR-3-COLOR. Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?



YES instance.

RSA public key cryptosystem (ch 31.7)

- Encrypt messages sent between 2 communication parties, an eavesdropper can not decode the message
- Relies on primes and the dramatic difference between:
  - Finding a large prime number (easy)
  - Factoring the product of two large prime numbers (hard)

If P = NP…

Mathematicians would be out of a job

Cryptography as we know it would not be possible (e.g. RSA)

AI program become perfect as exhaustive search is efficient

Various Problems

Hard problems (**NP**-complete)	Easy problems (in **P**)
3-SAT	2-SAT
Hamilton Path	Euler Path
Longest Path	Shortest Path
0-1 Knapsack	Fractional Knapsack
Traveling Salesperson	Minimum Spanning Tree
Vertex Cover	Vertex Cover in trees
Independent Set	Independent Set in trees
Subset sum	Sorting
Clique	Bipartite matching
Factorization	Finding a large prime

Solve NP-complete problems

- Three ways to get around NP-completeness.

1. if inputs are small, an algorithm with exponential running time may be perfectly satisfactory.
2. isolate important special cases that we can solve in polynomial time.
3. approaches to find *near-optimal* solutions in polynomial time (either in the worst case or the expected case).

Approximation algorithms: give near-optimal solutions in polynomial time.