# P vs. NP


Simpsons: Treehouse of Horror VI

# Attribution

- These slides were prepared for the New Jersey Governor’s School course “The Math Behind the Machine” taught in the summer of 2012 by Grant Schoenebeck
- Large parts of these slides were copied or modified from the previous years’ courses given by Troy Lee in 2010 and Ryan and Virginia Williams in 2009.

# Questions?

facebook

You are given the complete graph of Facebook.

What questions would you ask?

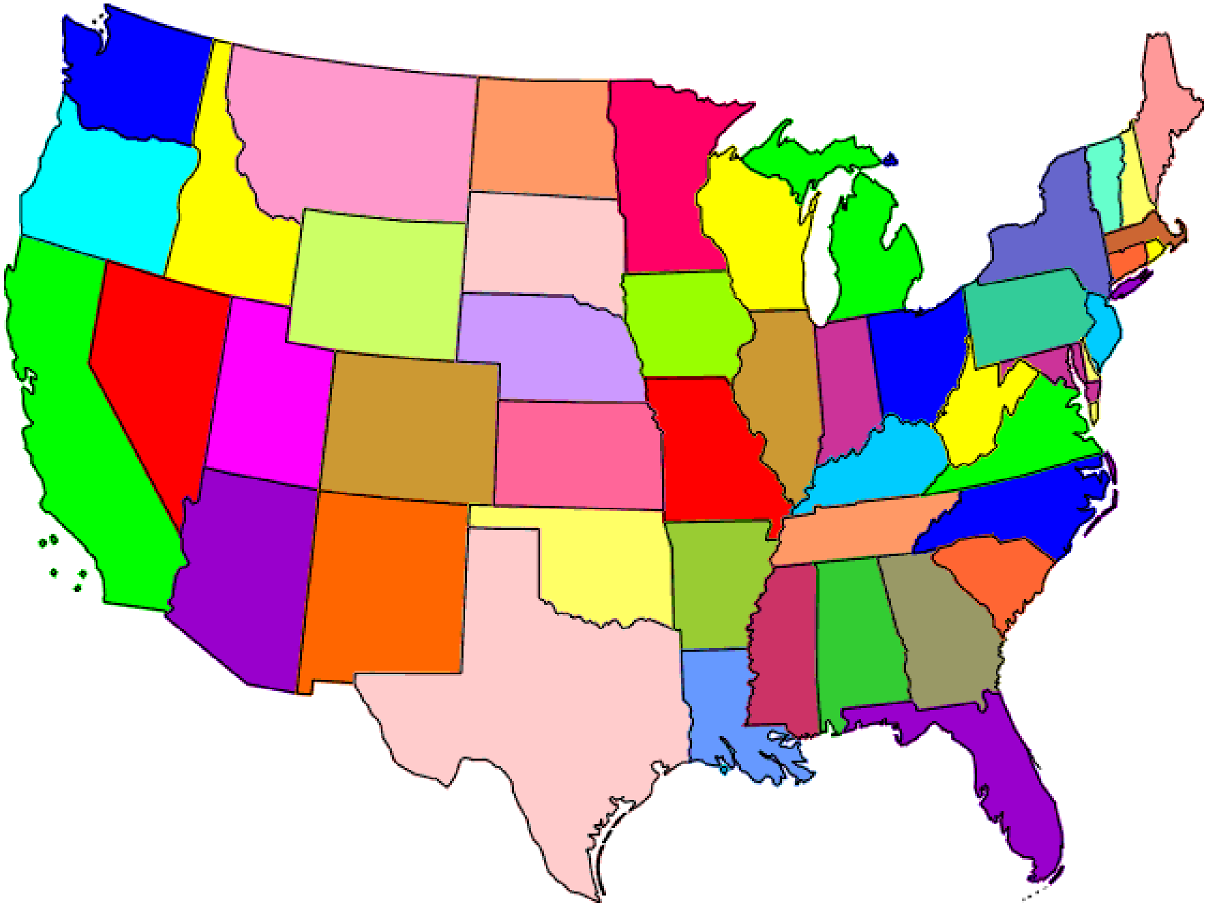
(What questions could we hope to answer?)

# What is NP?

Time for coloring...

# Coloring Maps

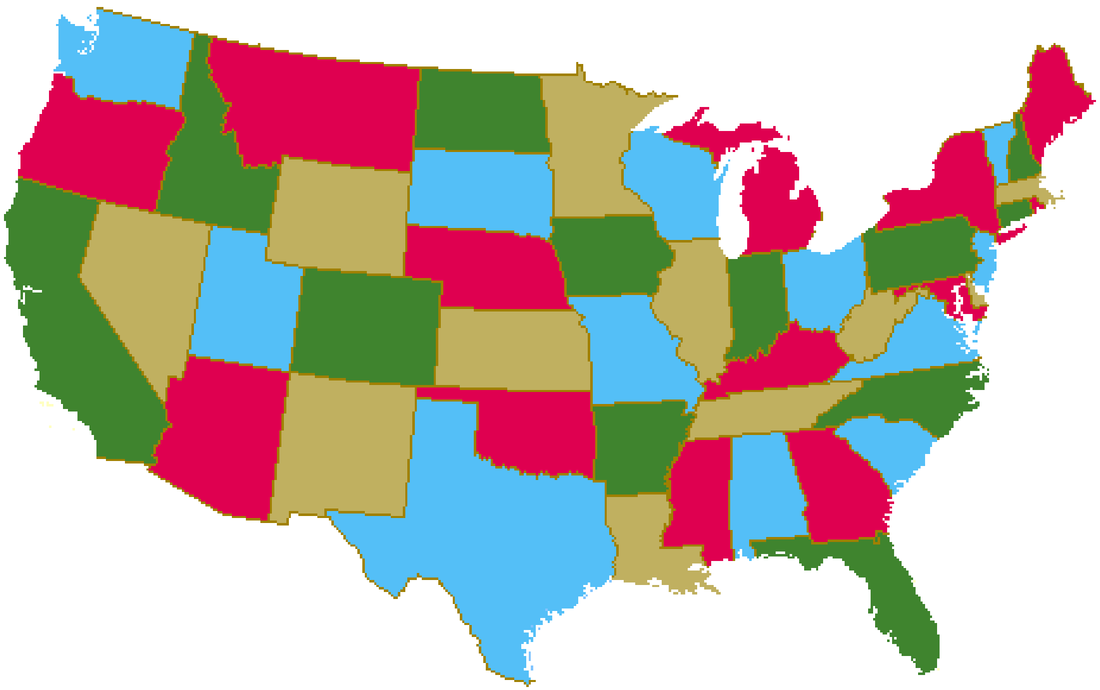
Try It!



In a map, don't want neighboring states to be the same color.
How many colors are needed?

# Coloring Maps

How many colors did you need?



# Are four colors necessary?

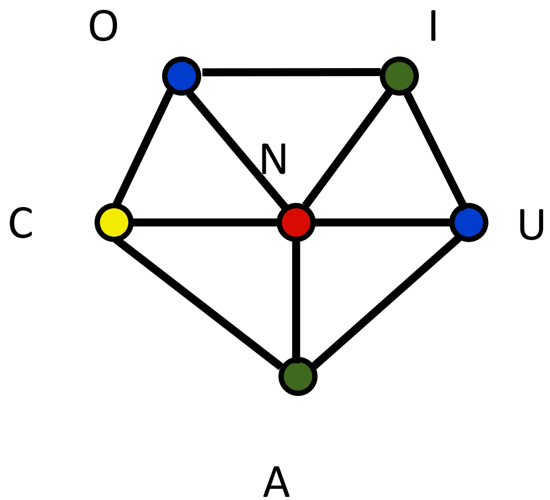
Look at the wild west…


Nevada has 5 nbhs.

The cycle of nbhs: oregon, idaho, utah, arizona, cali,oregon
requires 3 colors.

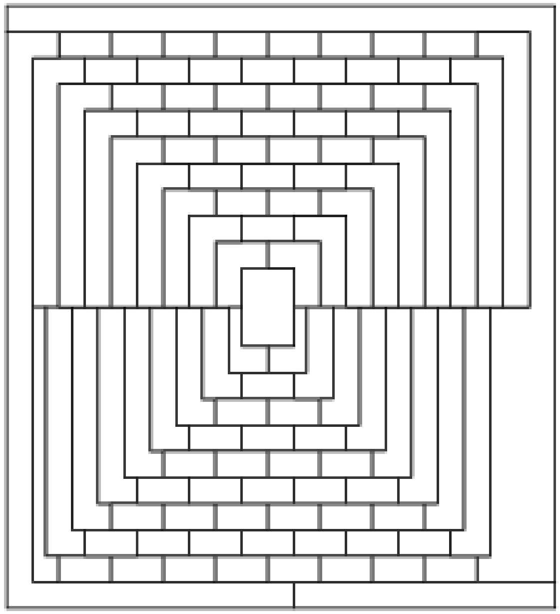
# Are four colors necessary?

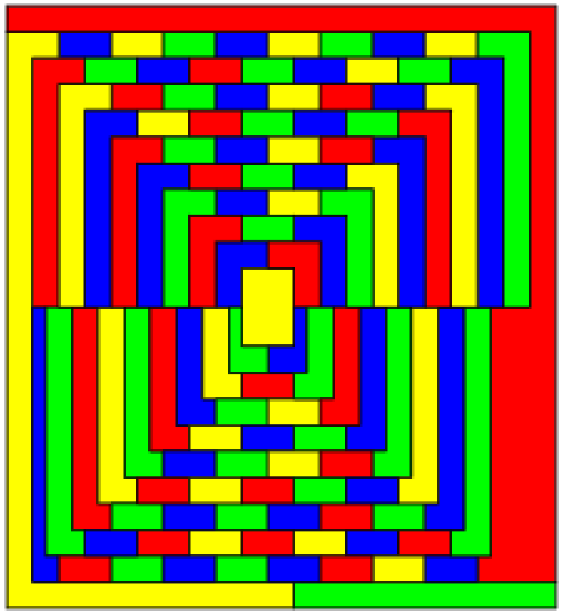
Look at the wild west...

# Four Color Theorem

Every map can be colored with only four colors.





# History of the four color theorem

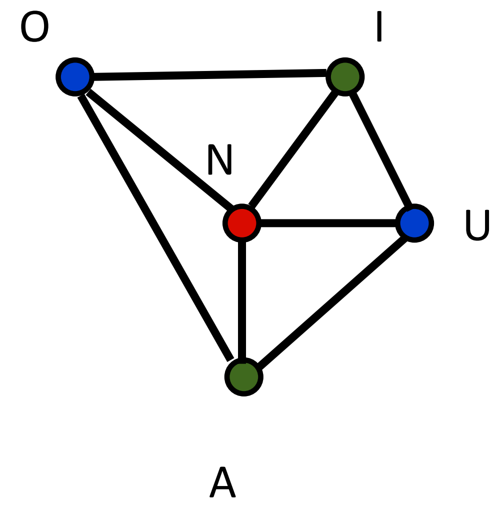
- 1852: Conjecture raised by Guthrie who noticed counties of England could be colored with only four colors.
- 1879: Proof given by Kempe.
- 1890: Heawood discovers flaw in Kempe’s proof. Proves that five colors suffice.
- 1976: Appel and Haken prove it. Proof relies heavily on computer verification. Reduce problem to examining 1,936 special maps.

# History of the four color theorem

- 1996: Sanders, Seymour, and Thomas give a simplified proof only requiring to check (by computer) 633 special maps.
- Their proof gives a $n^2$ time algorithm for four coloring a map.
- Still an open question to find a completely hand checkable proof!

# Where we stand

- Every map can be colored with four colors.
- Some can be colored with three.




- Some can be colored with two.  Can you tell which?

# 3-coloring problem

- Given a map, output “yes” if it can be colored with three colors, “no” otherwise.
- What is one possible algorithm to do this?
  - One could simply try all possible $3^n$ colorings. If a coloring “works,” it is easy to tell. Why coloring?
- Actually useful if:
  - Want to avoid conflicts.
  - Clustering by dissimilarity.

# The Sixties

- Many different routing and scheduling problems did not have efficient solutions.
- They all seemed difficult but different nuts to crack.

# NP

- **NP** is the class of problems which have solutions that can be efficiently verified.
  - As usual, efficiently means polynomial in size of input.
- **NP** stands for nondeterministic polynomial time.
- 3-coloring is in NP.  Given a proposed coloring, we can quickly check if it works.

# **P** vs. **NP**

- **P**: problems which we can efficiently solve.
- **NP**: problems which, given a proposed solution, we can efficiently check if it works.
- Every problem in **P** is also in **NP**.
- It is conjectured that there are problems in **NP**, for example 3-coloring, that are not in **P**.

# **P** vs. **NP**

- Figuring out the relationship between P and NP is one of, if not the, greatest open problem in mathematics today.

- It is one of the 6 ~~7~~ open problems which the Clay Mathematics institute is offering $1,000,000 for its resolution.

# Importance of **P** vs. **NP**

- Given a conjecture, find a proof.
- Given data on some phenomenon, find a theory explaining it.
- Given constraints (cost, strength, energy), find a design (bridge, medicine, phone).
- Code breaking.
- In each of these cases, when we see a good solution, we should be able to recognize it!

# **NP**-completeness

- Around 1971, Cook and Levin developed the idea of **NP**-completeness. Soon after Karp showed 24 **NP**-complete problems off all shapes and colors.
- These are universal **NP**-problems...if you can solve them efficiently, you can solve ANY problem in **NP** efficiently.
- L is **NP**-complete if:
  – L is in **NP**
  – ANY other problem in NP reduces to L.
- If you come up with an efficient algorithm to 3-color a map, then **P**=**NP**.

# NP-completeness

- Today over 3000 NP-complete problems known across all the sciences.
- Google Scholar search of NP-complete and biology returns over 10,000 articles.

# NP-completeness

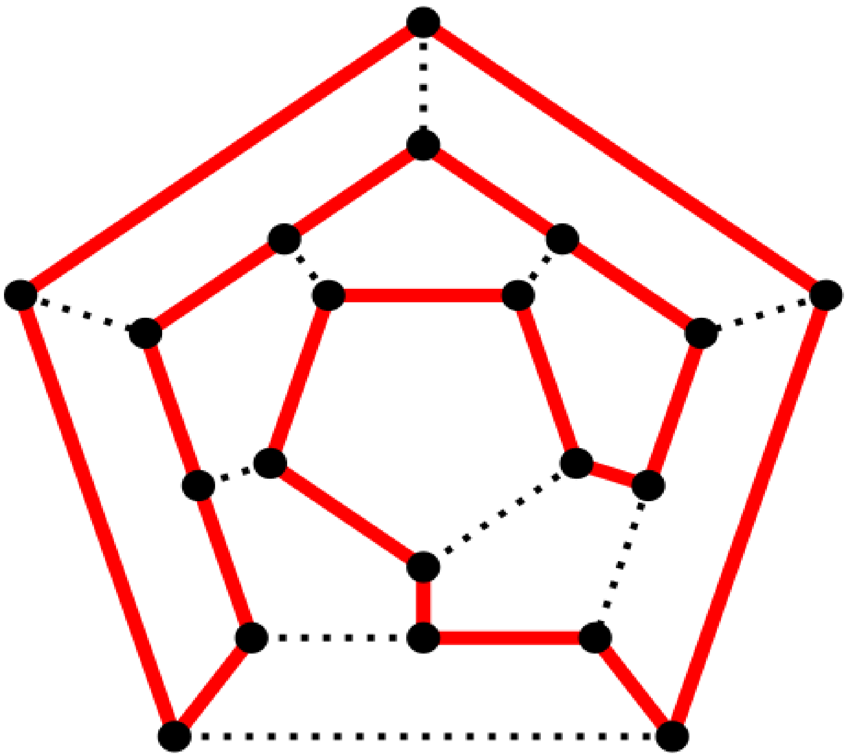
- Quintessential NP complete problem: 3 SAT.
- $(x_1 \vee \neg x_4 \vee x_5) \wedge (\neg x_1 \vee x_6 \vee \neg x_5) \wedge \cdots$
- Given a formula like this, does it have a Boolean assignment which makes it true?

# Integer Programming

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \quad & \leq b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \quad & \leq b_2 \\
& \vdots \\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n \quad & \leq b_n \\
x_1, x_2, \ldots, x_n \quad & \in \{0, 1\}
\end{aligned}
$$

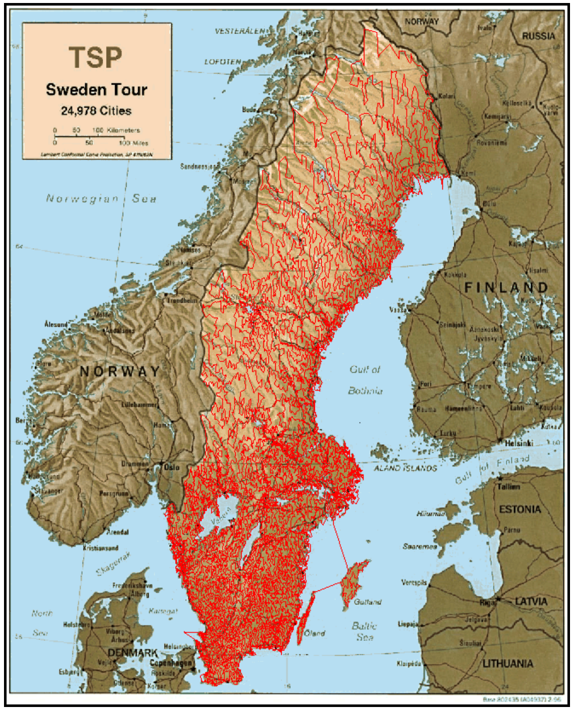
# Hamiltonian Cycle

Given a graph, is there a cycle which visits all vertices exactly once?



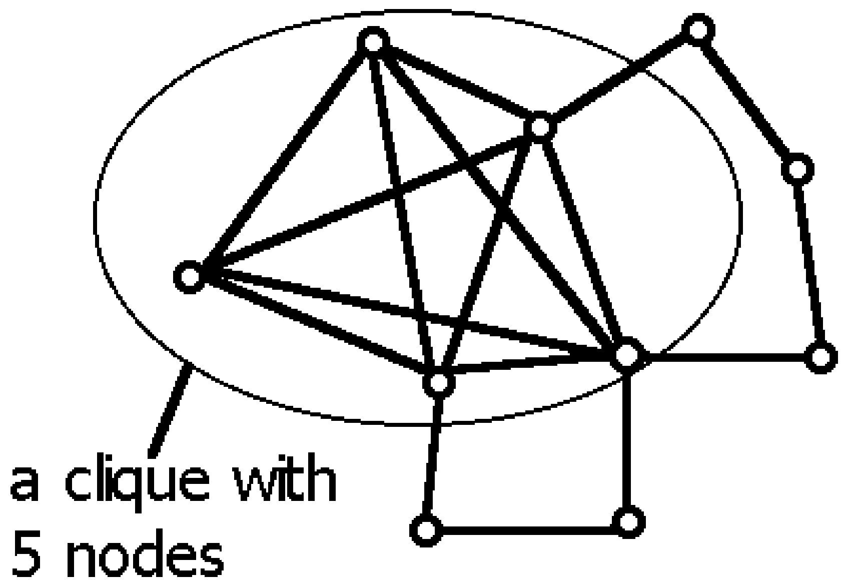
# Traveling Salesman Problem

Given a list of cities and pairwise distances between them, is there a tour which visits each city exactly once and has length at most k?



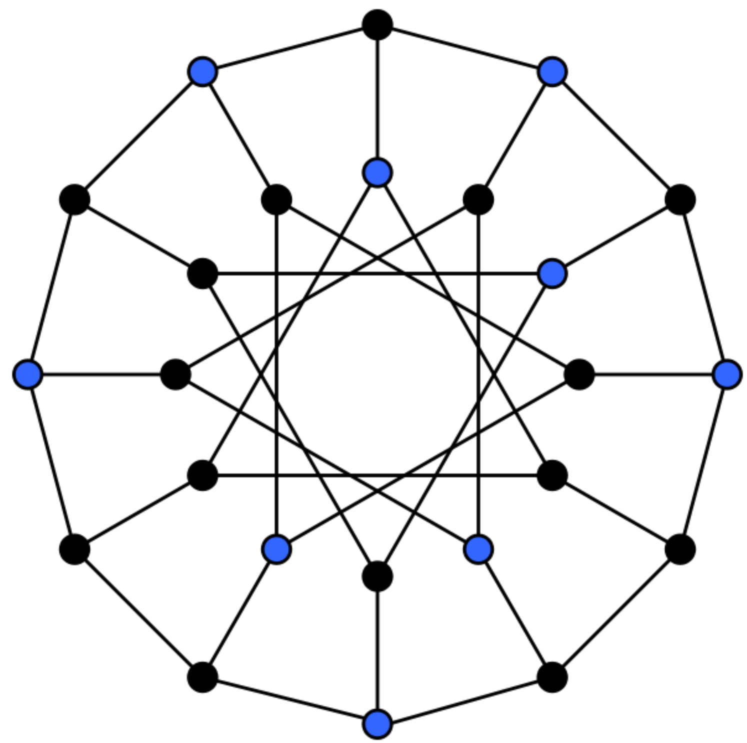
# Clique

- Given (G, k) a graph and integer: Are there k nodes in G that are all connected to each other?



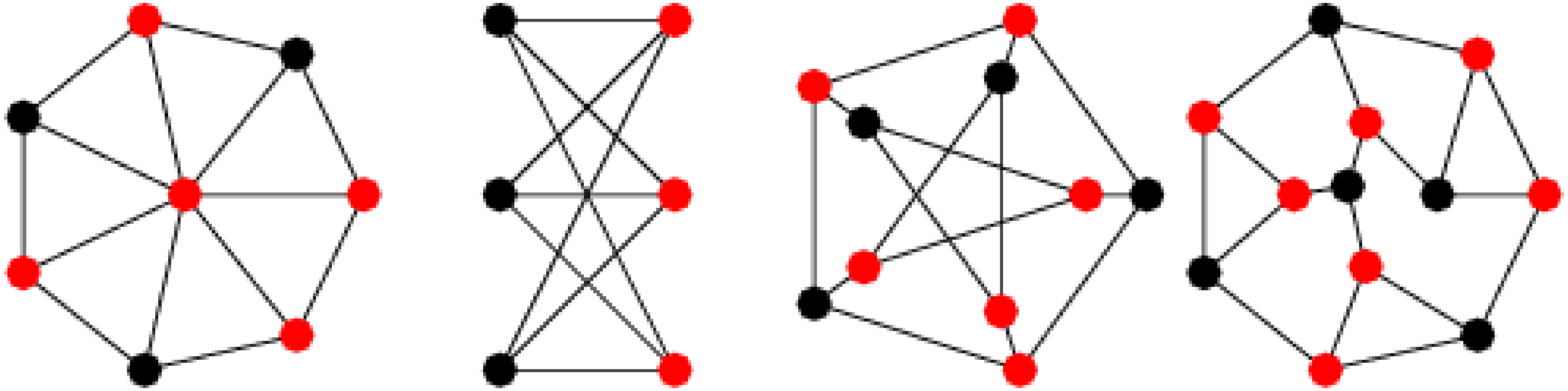
# Independent Set

- Given (G, k) a graph and integer, are there k nodes in G none of which are connected to each other.



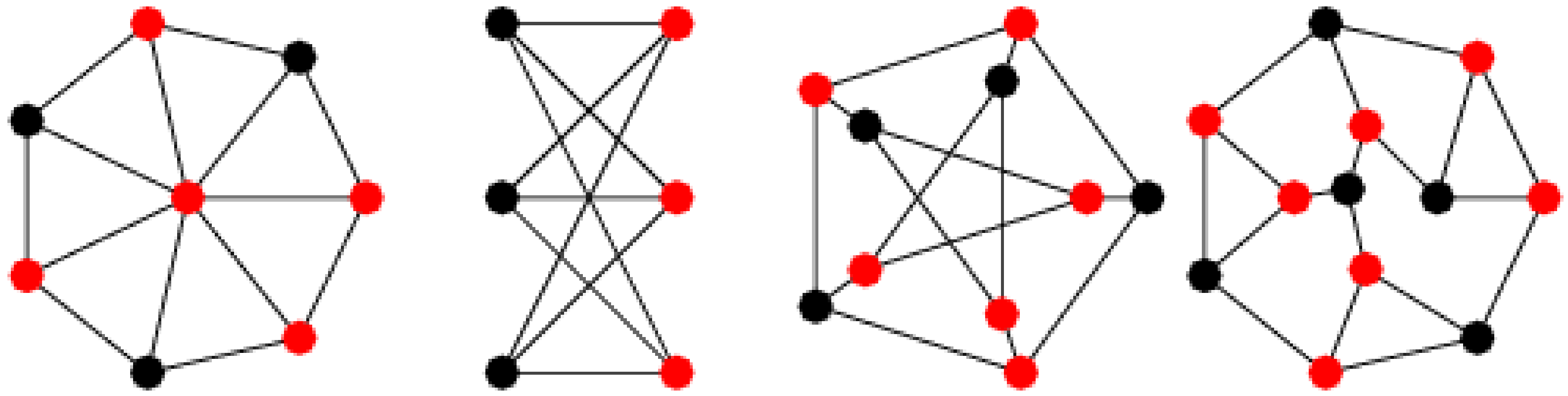
# Vertex Cover

- Given (G, k) a graph and integer, are there k nodes in G none of which are connected to each other.



# Set Cover

- Given list of items S and subsets of S: $S_1$, $S_2$, …, $S_m$ and integer k: are there k subsets that "cover" S?
- Vertex Cover is a special case where each element is in exactly 2 sets.



# **NP** reductions

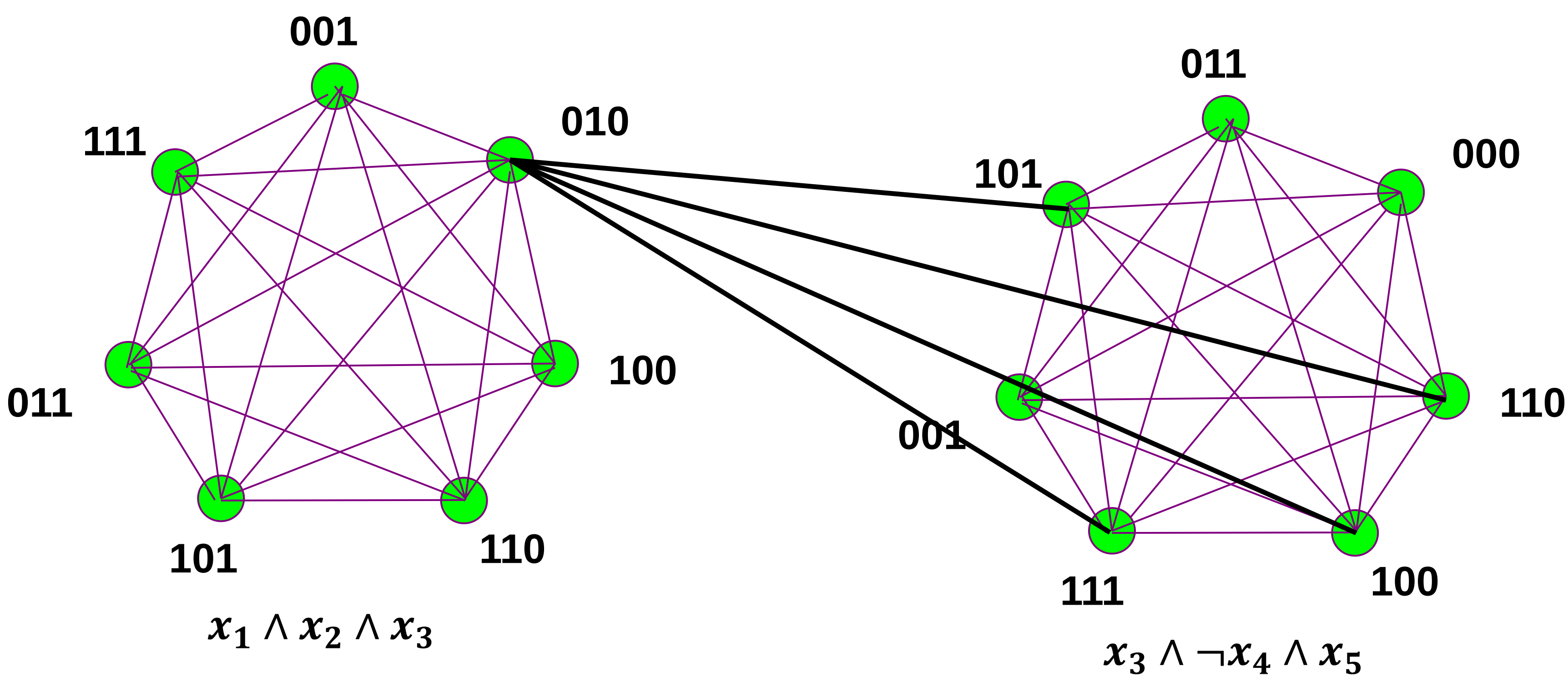
- 3-color ⇒ 3-SAT ⇒ Independent Set ⇒ Clique
- Independent Set ⇒ Vertex Cover ⇒ Set Cover
- Hamiltonian Cycle ⇒ Traveling Salesman Problem
- Idea: Given **NP**-complete problem A; problem B in **NP**. If A reduces to B than *every* **NP**-problem reduces to B. Therefore B is **NP**-complete.
- Implementation: Write an efficient procedure for A assuming a subroutine that solves B.

# 3-coloring to 3-SAT

- Given a 3-coloring problem G = (V, E), create 3-SAT problem with:
- Variables
  $x_{(v,\ c)}\ \forall\ v \in V,\ c \in \{g,\ r,\ b\}$
- Clauses
  - $\forall (v,u) \in E\ c \in \{g,\ r,\ b\}$
    $\neg\ x_{(v,\ c)} \vee \neg\ x_{(u,\ c)}$
  - $\forall\ v \in V$
    $\neg\ x_{(v,\ g)} \vee \neg\ x_{(v,\ r)} \vee \neg\ x_{(v,\ r)}$
    $\neg\ x_{(v,\ g)} \vee \neg\ x_{(v,\ b)} \vee \neg\ x_{(v,\ b)}$
    $\neg\ x_{(v,\ b)} \vee \neg\ x_{(v,\ r)} \vee \neg\ x_{(v,\ r)}$
    $x_{(v,\ g)} \vee x_{(v,\ r)} \vee x_{(v,\ b)}$
- 3-coloring: color vertices of graph g, r, b so that adjacent edges do not have same color
- 3-SAT: is there an assignment simultaneously satisfying all clauses

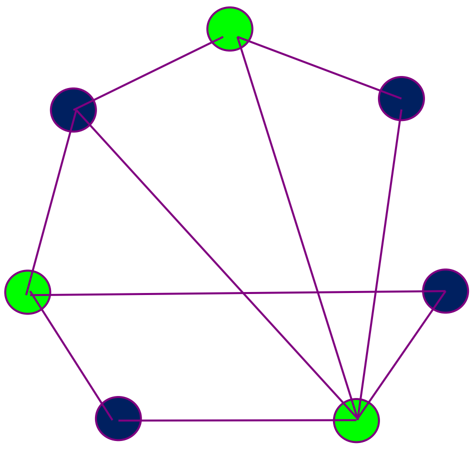
# 3-SAT to Independent Set

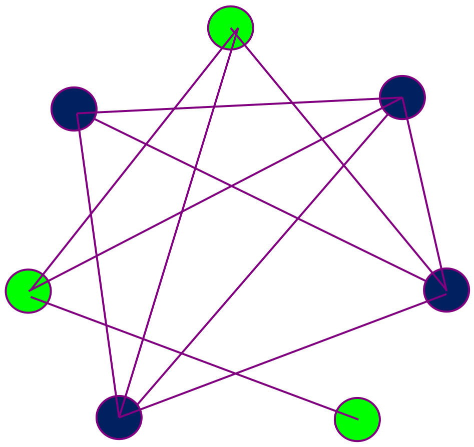
- Write procedure for 3-SAT given a subroutine computing Independent Set (G, k)
  - 3-SAT: is there an assignment simultaneously satisfying all clauses
  - Independent Set: Given (G, k) a graph and integer, are there k nodes in G none of which are connected to each other.



# Independent Set to Clique

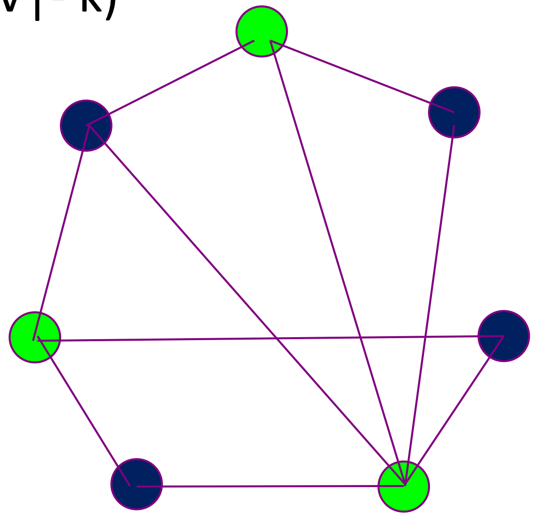
- Write an efficient procedure IndependentSet(G, k) given subroutine Clique(G', k'):
  - Independent Set: Given (G, k) a graph and integer, are there k nodes in G none of which are connected to each other.
  - Clique: Given (G, k) a graph and integer: Are there k nodes in G that are all connected to each other?
- Ask Clique (G', k) where G' is G with "opposite" edges.

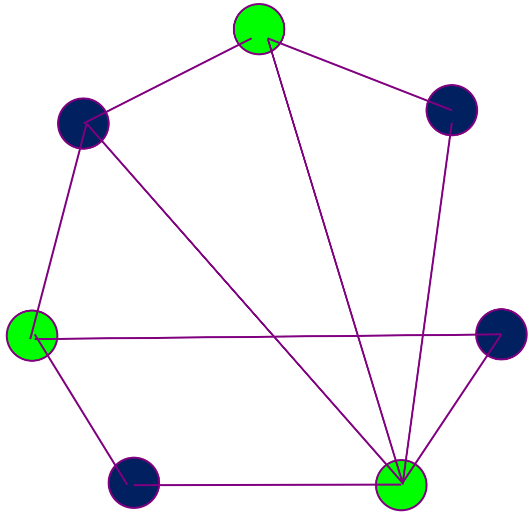




# Independent Set to Vertex Cover

- Write an efficient procedure IndependentSet(G, k) given subroutine VertexCover (G’, k’):
  - Independent Set: Given (G, k) a graph and integer, are there k nodes in G none of which are connected to each other.
  - VertexCover: Given (G, k) a graph and integer: Are there k nodes in G that are incident on all edges?
- Given IndependentSet problem (G, k)
  Ask Vertex Cover (G, |V|- k)





# Vertex Cover to Set Cover

- Write an efficient procedure VertexCover(G, k) given subroutine SetCover(S, $S_1$,…, $S_m$, k'):
  - VertexCover: Given (G, k) a graph and integer: Are there k nodes in G that are incident on all edges?
  - SetCover: Given list of items S and subsets of S: $S_1$, $S_2$, …, $S_m$ and integer k: are there k subsets that "cover" S?

- Given VertexCover problem (G = (V, E), k), let S = E, $\forall v \in V\ let\ \ S_v = \{e \in E: e = (v,*)\}$, k' = k

# Clique to ½-Clique

- Write an efficient procedure Clique(G, k) given subroutine ½-Clique(G).
  - ½ - Clique (G): Does G have |V|/2 nodes that are all connected to each other?
  - Clique(G, k): Given (G, k) a graph and integer: Are there k nodes in G that are all connected to each other?
- Given Clique problem (G, k) create ½-clique problem G' where you create |V| new vertices and attach |V|-k of them to every other vertex (new and old).

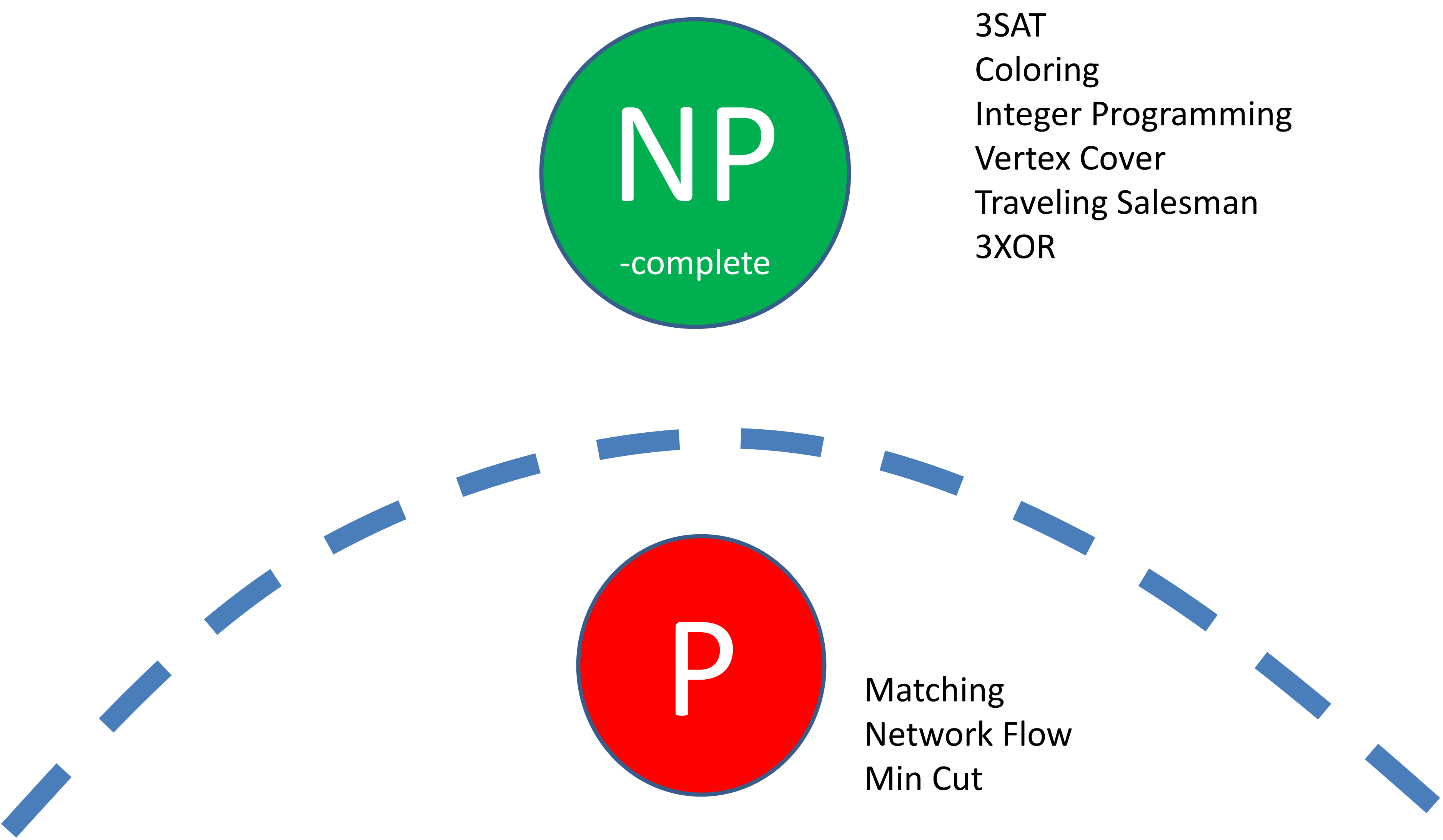
# Hamiltonian Cycle to TSP

- Write a program that solves Hamiltonian Cycle
- You are have a subroutine TSP(all pair distances D, int k) that reports the solution to TSP
  - Hamiltonian Cycle: given a graph, is there a cycle which visits all vertices **exactly** once?
  - TSP: Given a list of cities and pairwise distances between them, is there a tour which visits each city exactly once and has length at most k?
- Given G = (V, E) create D where distance between u and v is
  - 1 if $(u,v) \in E$
  - 2 if $(u,v) \notin E$
- Report back TSP(D, $|V|$)

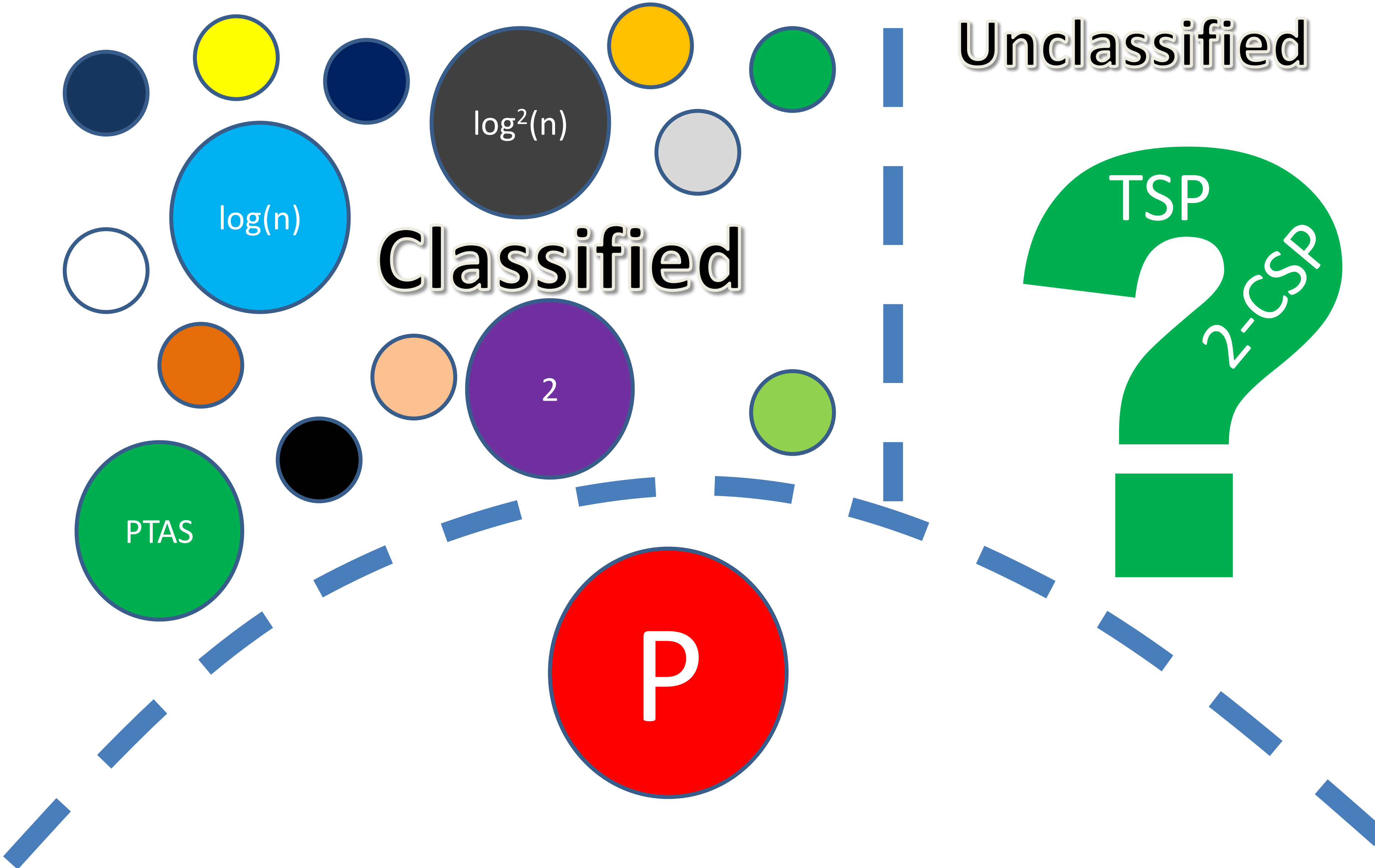
# So What Now?

- Relax worst case
  - Use CPLEX/SAT solver
- Relax time constraints
- Relax exact optimal
  - Can we closely approximate?

# Combinatorial Optimization Problems



# Approx Combinatorial Optimization



# Proving coNP statements

- Resolution $(x \vee A) \wedge (\neg x \vee B) \Rightarrow (A \vee B)$
- Can try to take 3-SAT formula and resolve!
- Thm: any false statement will resolve to

$$(x) \wedge (\neg x)$$

- Thm: “RANDOM” 3-Sat will take time $2^n$ to resolve.
- Phases of hardness

# Approximation Algorithms

- Cannot solve Vertex Cover, but can we find a good approximation for it?
- Recall VertexCover: Given (G, k) a graph and integer: Are there k nodes in G that are incident on all edges?
- Can we approximate it?
- Wait! That would require us to prove an approximate coNP statement: that the VertexCover is not too small.

# Approximating Vertex Cover

- Recall VertexCover: Given (G, k) a graph and integer: Are there k nodes in G that are incident on all edges?
- A maximal matching M is a set of edges such that
  - Each vertex is incident on at most one edge in M
  - No edge can be added without violating first condition
- Let M be Maximal matching and S any minimum vertex cover: then $|M| \leq |S| \leq 2|M|$
  - $|M| \leq |S|$ because $(\mathrm{u}, \mathrm{v}) \in M \Rightarrow u \in S \; or \; v \in S$
  - $|S| \leq 2|M|$ because the vertices of M are a vertex cover.

# Approximating Vertex Cover II

min $\sum_{\{v \in V\}} x_v$

$\forall\ (u, v) \in E$ $\qquad x_v + x_u \geq 1$

$\forall v \in V$ : $\qquad \cancel{x_v \in \{0, 1\}}$ $\qquad 0 \leq x_v \leq 1$

Integrality Gap = $\dfrac{\text{IP}}{\text{LP}}$

$\geq$ 2 by rounding

$\leq$ 2: complete graph

IP = n-1

LP = n/2

# Greedy Set Cover

- SetCover: Given list of n items S and subsets of S: $S_1, S_2, \ldots, S_m$ and integer k: are there k subsets that "cover" S?
- Take largest set, then largest set with respect to remaining elements and repeat as necessary.
- At each step, greedy covers at least a 1/OPT fraction of what is left. So takes at most x steps where $n \cdot \left(1 - \frac{1}{OPT}\right)^x < 1$ so x = O(log(n))*OPT