Hard vs Easy Problems

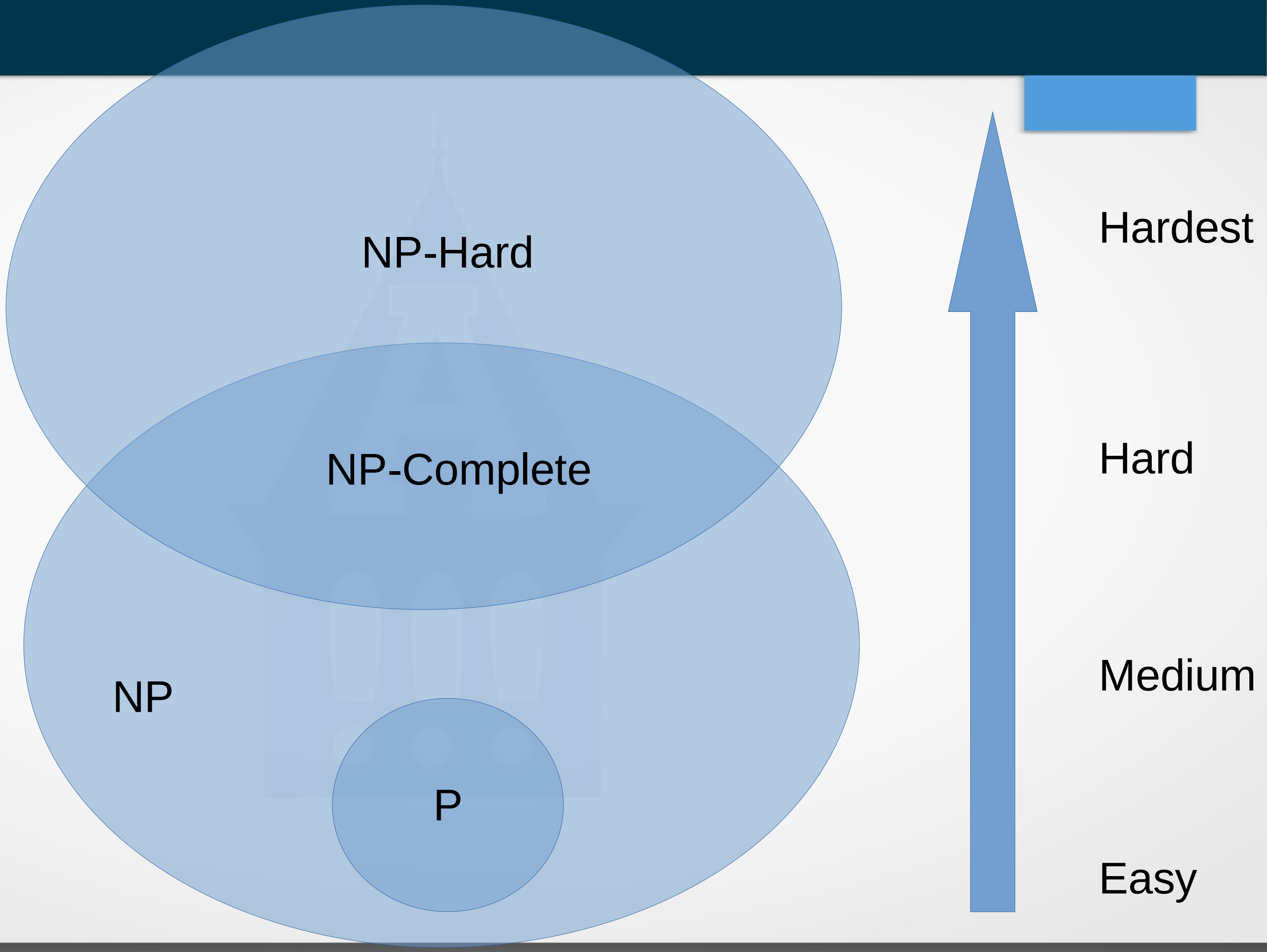
Hard vs Easy Problems

- Informally…
  - ***Easy*** is polynomial in the problem size
  - ***Hard*** is exponential in the problem size

Some Terminology

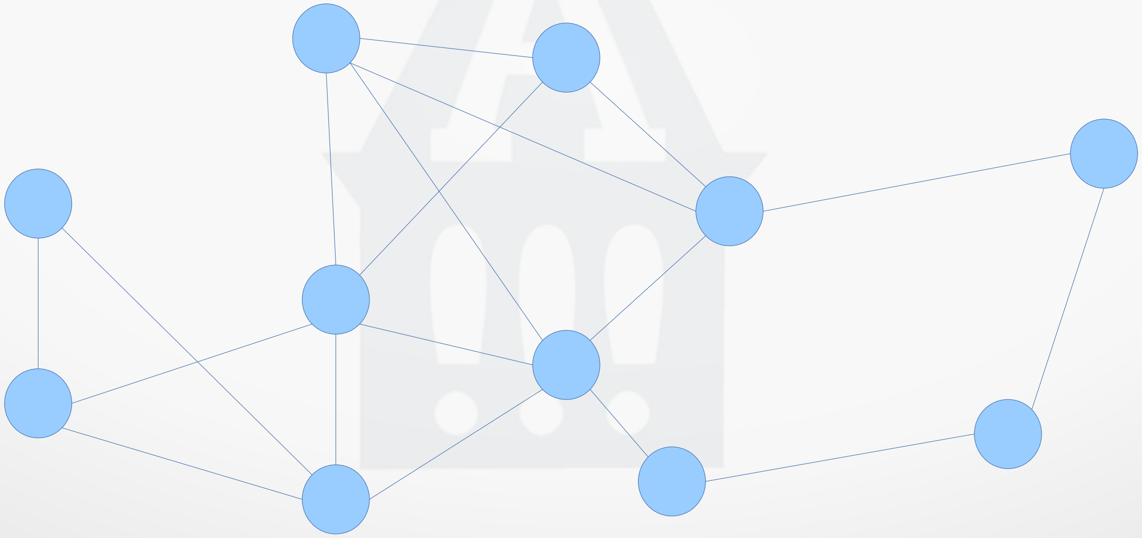
- **NP**
  - NP is the set of all decision problems (questions with yes-or-no answer) for which the ‘yes’ answers can be ***verified*** in polynomial time $O(n^k)$; where n is the problem size and k is a constant.
  - Polynomial time is sometimes used as the definition of *fast* or *quickly*
  - Examples – verification of: Linear Search O(n), Insertion/any Sort O(n)
- **P**
  - P is the set of all decision problems which can be ***solved*** in polynomial time.
  - Since it can be solved in polynomial time, it can also be verified in polynomial time. Therefore, P is a subset of NP
  - Examples: Showing a number is prime $O(n \log^{21/2})$
- **NP-Complete**
  - Can’t find an ***efficient*** algorithm to ***solve*** (a polynomial algorithm)
  - The complexity suggests the problem is intractable – exponential
  - Terminology
    - intractable, unmanageable, uncontrollable, out of hand, impossible to cope with
  - Example: Hamiltonian Cycle
- **NP-Hard**
  - Both hard to solve and hard to verify, possibly not even decidable
  - Examples: Traveling Salesperson, Vertex Cover

Some Terminology



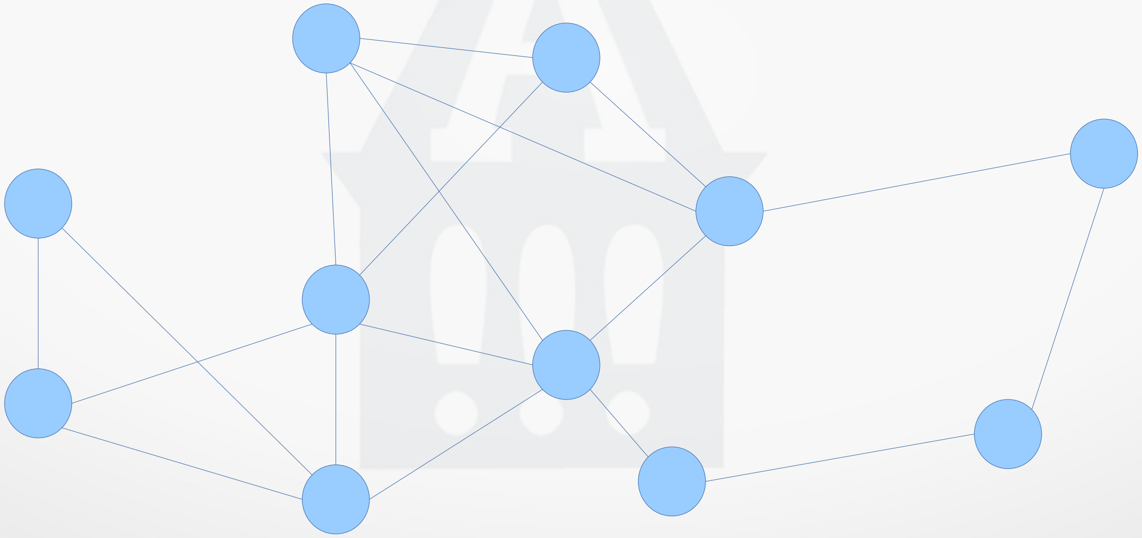
Hard Problem – Traveling Salesperson

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city (once) and returns to the origin city?
  - Sounds like Hamiltonian: It is similar, but we are additionally asking for it to be the shortest distance



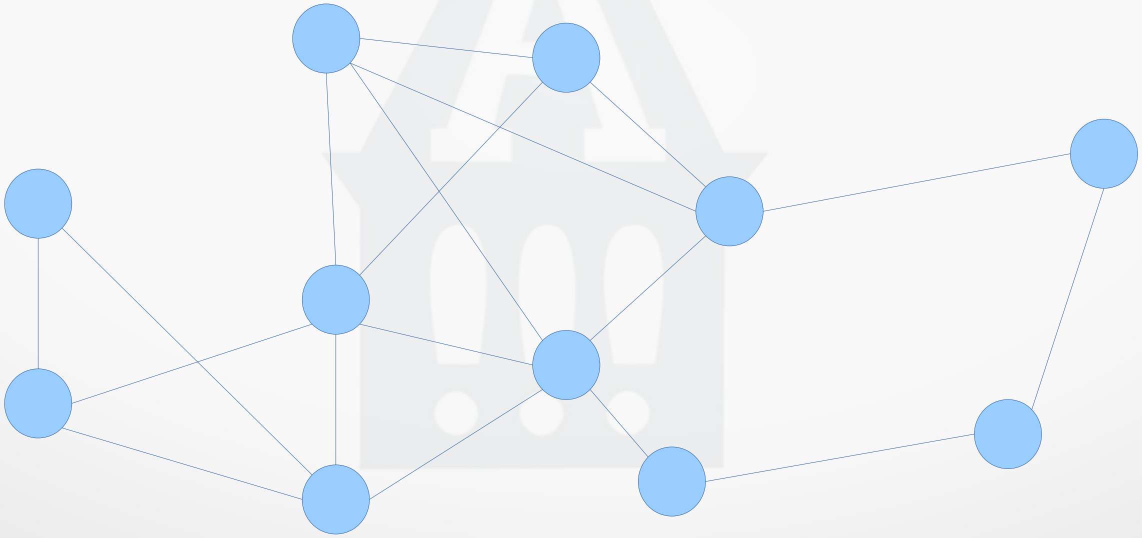
Hard Problem – Traveling Salesperson

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city (once) and returns to the origin city?
  - Sounds like Hamiltonian: It is similar, but we are additionally asking for it to be the shortest distance
- Have to try all possible combinations, $O(n\ 2^n)$\{n times 2 to the n\}, ouch!



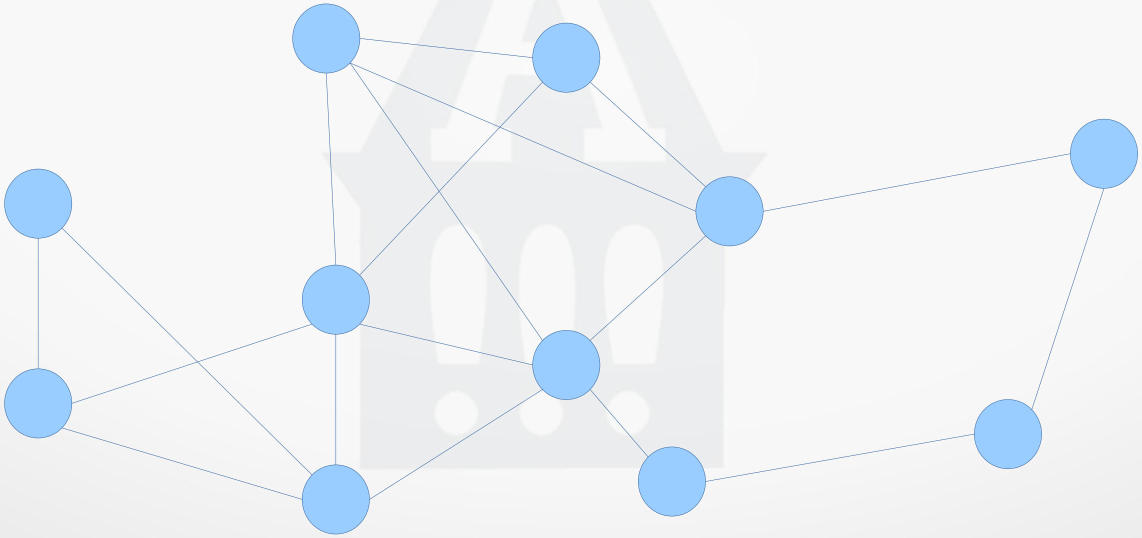
Hard Problem – Find Largest Complete Subgraph (clique)

- Used in molecular biology to find common structures



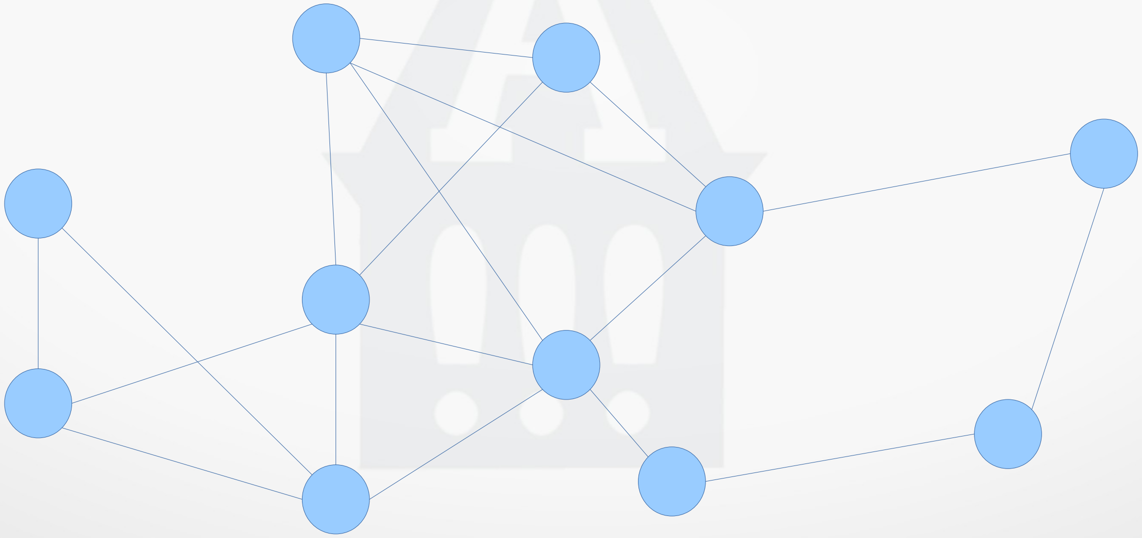
Hard Problem – Find Largest Complete Subgraph (clique)

- Used in molecular biology to find common structures
- Have to try all possible subsets of the nodes, $O(2^n)$, ouch!



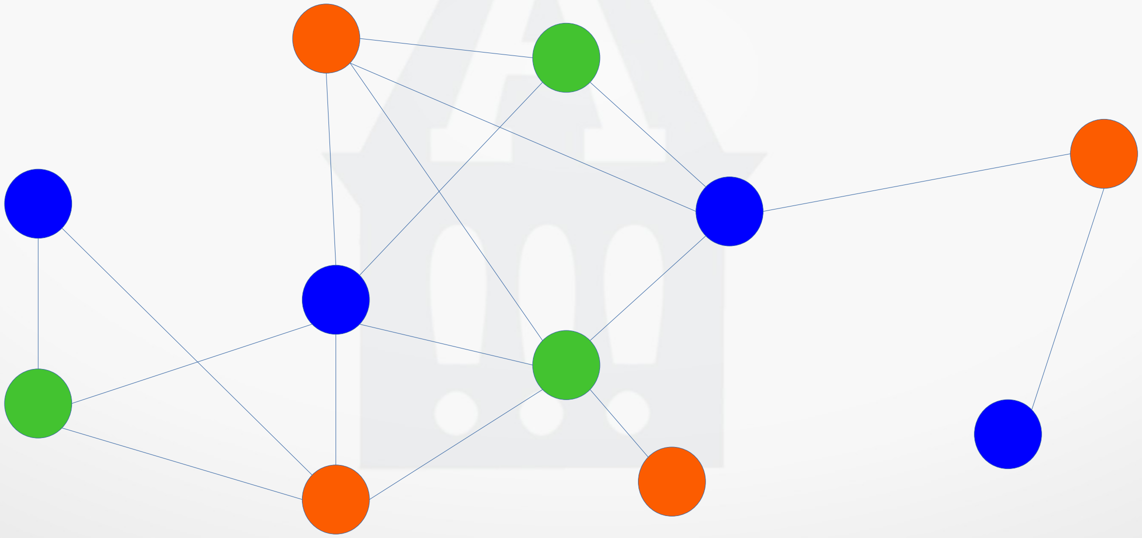
Hard Problem – Hamiltonian Path

- Yep, its hard!



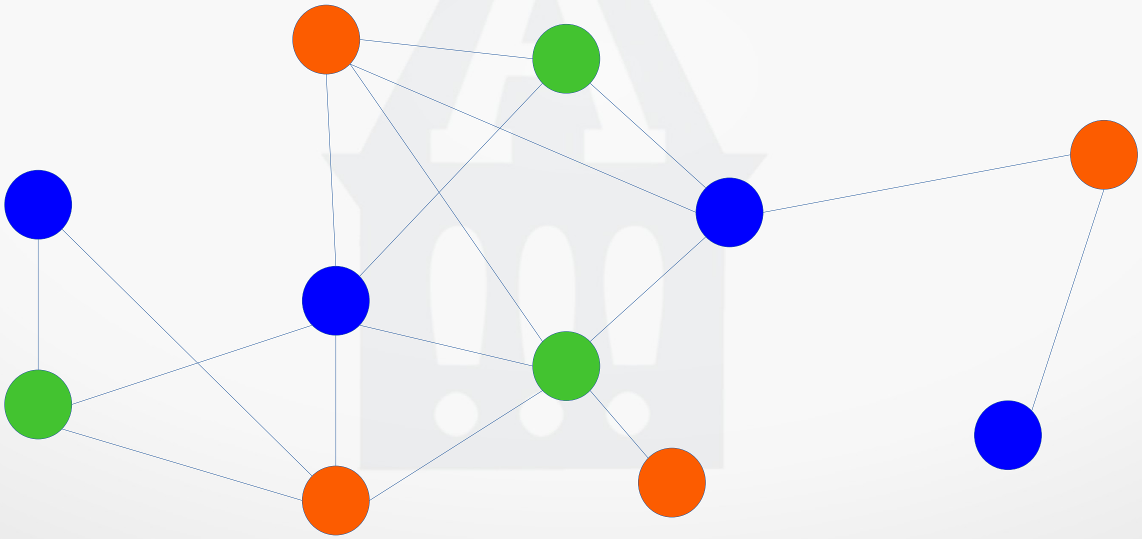
Hard Problem – Graph Coloring

- Color the vertices of a graph (in the minimum number of colors) such that no two adjacent vertices are of the same color



Hard Problem – Graph Coloring

- Color the vertices of a graph (in the minimum number of colors) such that no two adjacent vertices are of the same color
- Yep, its hard!



Hard Problem – Graph Coloring

- Why do we care about graph coloring?

- We need to find a time for various clubs to meet…
  - such that individuals who are in multiple clubs can attend all meetings
  - would like to use the minimal number of meeting times possible

- What are the nodes, edges, colors?

Hard Problem – Graph Coloring

- Why do we care about graph coloring?

- We need to find a time for various clubs to meet…
  - such that individuals who are in multiple clubs can attend all meetings
  - would like to use the minimal number of meeting times possible

- What are the nodes, edges, colors?
  - nodes: clubs
  - edges: common member (person) between clubs
  - colors: meeting times

Hard Problem – Graph Coloring

- Pair Programming Problem
  - With N students and K projects, where N is even, can we:
    - Assign pairs of students to each project
    - Every student works on every project
    - No student has the same partner more than once
  - Can this be phrased as a graph problem? (I probably wouldn’t be asking if it couldn’t)

Hard Problem – Graph Coloring

- Pair Programming Problem
  - With N students and K projects, where N is even, can we:
    - Assign pairs of students to each project
    - Every student works on every project
    - No student has the same partner more than once
  - Can this be phrased as a graph problem? (I probably wouldn't be asking if it couldn't)
    - How about a graph coloring problem?

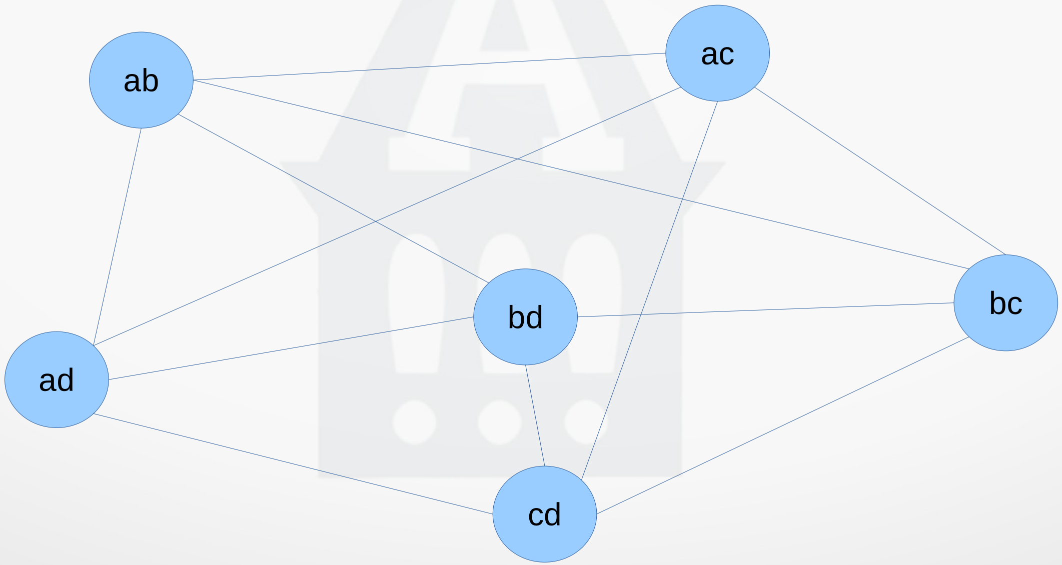
Hard Problem – Graph Coloring

- Pair Programming Problem
  - With N students and K projects, where N is even, can we:
    - Assign pairs of students to each project
    - Every student works on every project
    - No student has the same partner more than once
  - Can this be phrased as a graph problem? (I probably wouldn't be asking if it couldn't)
    - How about a graph coloring problem?

- Nodes are pairs of students
- Edges are "contains a common member"
- Colors are the projects

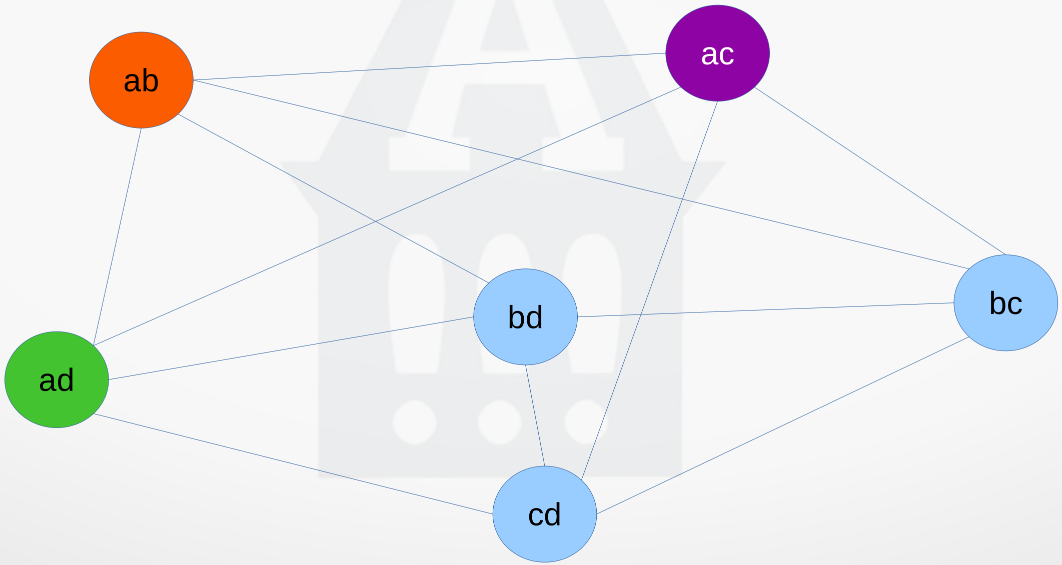
Hard Problem – Graph Coloring

- Pair Programming Problem
  - Nodes are pairs of students
  - Edges are “contains a common member”
  - Colors are the projects



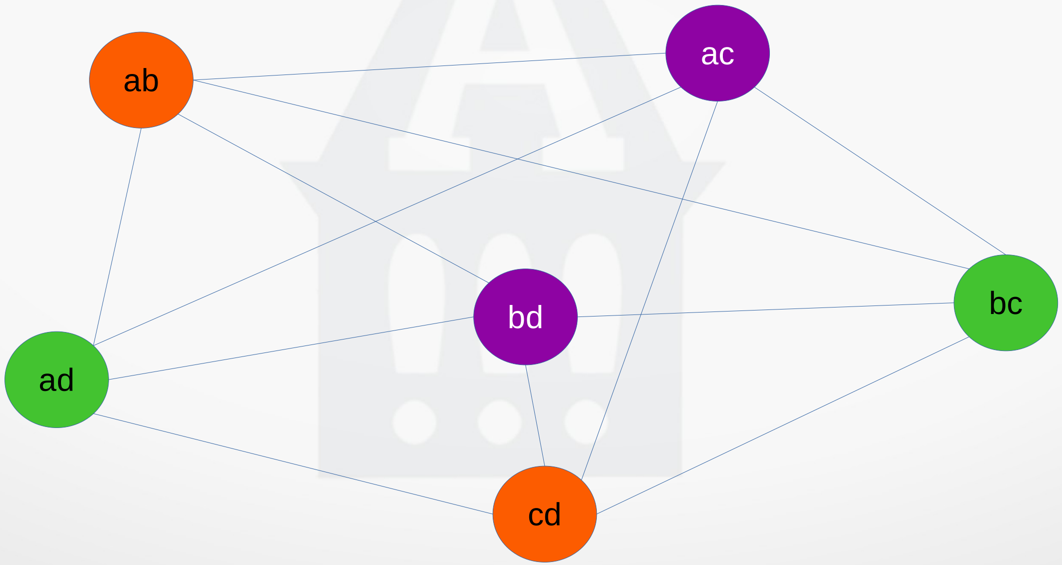
Hard Problem – Graph Coloring

- Pair Programming Problem
  - Nodes are pairs of students
  - Edges are “contains a common member”
  - Colors are the projects



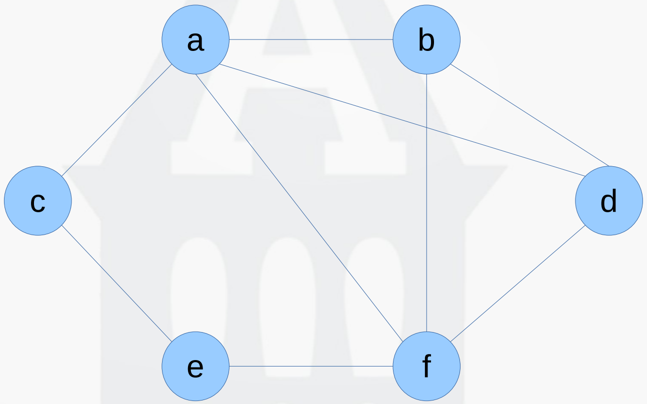
Hard Problem – Graph Coloring

- Pair Programming Problem
  - Nodes are pairs of students
  - Edges are “contains a common member”
  - Colors are the projects



Hard Problem – (Minimum) Vertex Cover

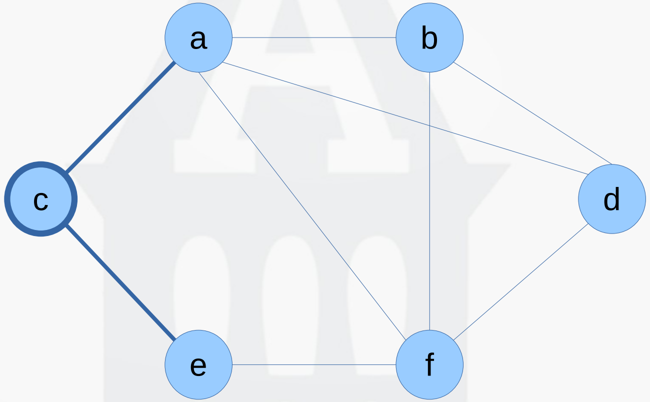
- Given a graph, is there is there a collection of k vertices such that each edge is connected to one of the vertices in the collection?



- A set of vertices that includes at least one endpoint of every edge of the graph

Hard Problem – (Minimum) Vertex Cover

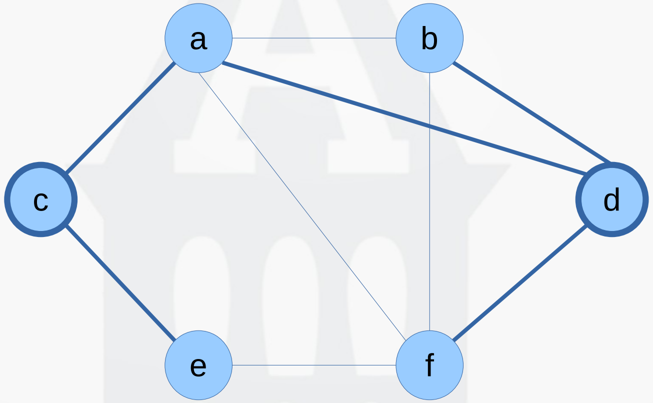
- Given a graph, is there is there a collection of k vertices such that each edge is connected to one of the vertices in the collection?
- Let's start with vertex c



- A set of vertices that includes at least one endpoint of every edge of the graph

Hard Problem – (Minimum) Vertex Cover

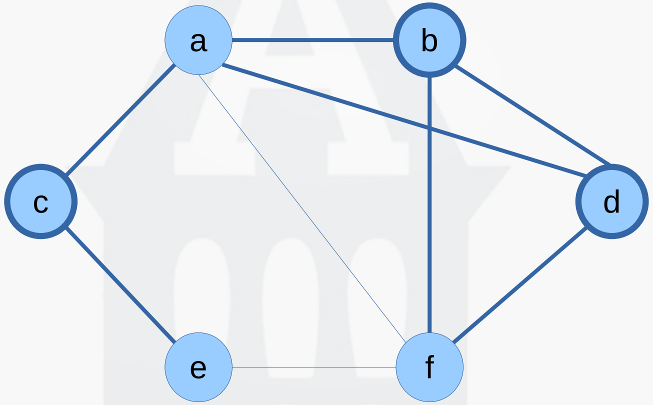
- Given a graph, is there is there a collection of k vertices such that each edge is connected to one of the vertices in the collection?
- Next, let’s add d



- A set of vertices that includes at least one endpoint of every edge of the graph

Hard Problem – (Minimum) Vertex Cover

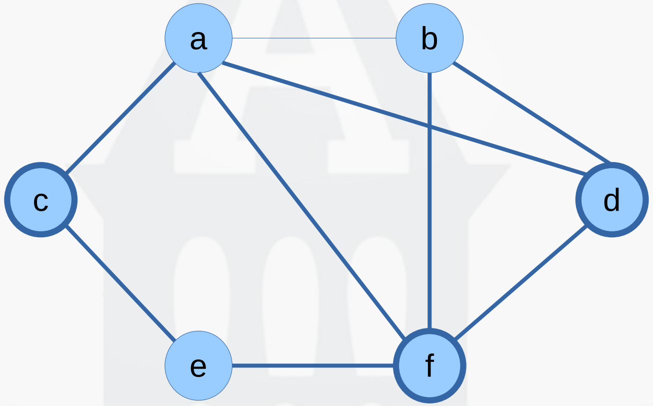
- Given a graph, is there is there a collection of k vertices such that each edge is connected to one of the vertices in the collection?
- Let’s try adding b



- A set of vertices that includes at least one endpoint of every edge of the graph

Hard Problem – (Minimum) Vertex Cover

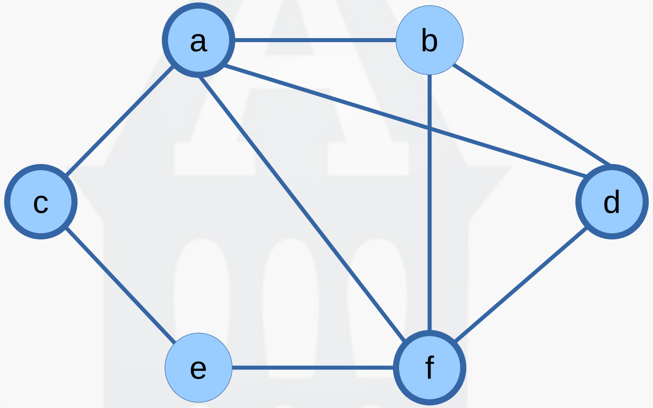
- Given a graph, is there is there a collection of k vertices such that each edge is connected to one of the vertices in the collection?
- That wasn't so good, let's try f instead



- A set of vertices that includes at least one endpoint of every edge of the graph

Hard Problem – (Minimum) Vertex Cover

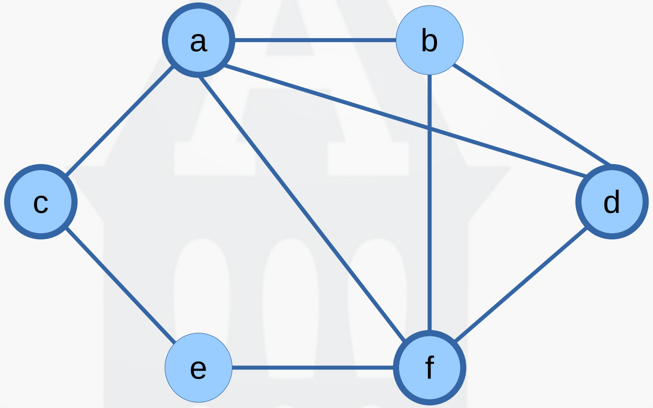
- Given a graph, is there is there a collection of k vertices such that each edge is connected to one of the vertices in the collection?
- Let’s add a; is that the best solution?
- Will a greedy solution always give the best?
- How do we know?



- A set of vertices that includes at least one endpoint of every edge of the graph

Hard Problem – (Minimum) Vertex Cover

- Given a graph, is there is there a collection of k vertices such that each edge is connected to one of the vertices in the collection?
- Yep, its hard!



- A set of vertices that includes at least one endpoint of every edge of the graph

Hard Problem – (Minimum) Vertex Cover – So What?

- Text summarization: The process of reducing the content of a document with an automated system that retains the most important points of the original document
  - We see the problem of summarization as a problem of selecting important sentences from a set of all sentences (the smallest set that connect all other ideas?)

- Design a (camera) security system in a building
  - Have a view of all hallways, entrances, etc.
  - Use the fewest number of cameras