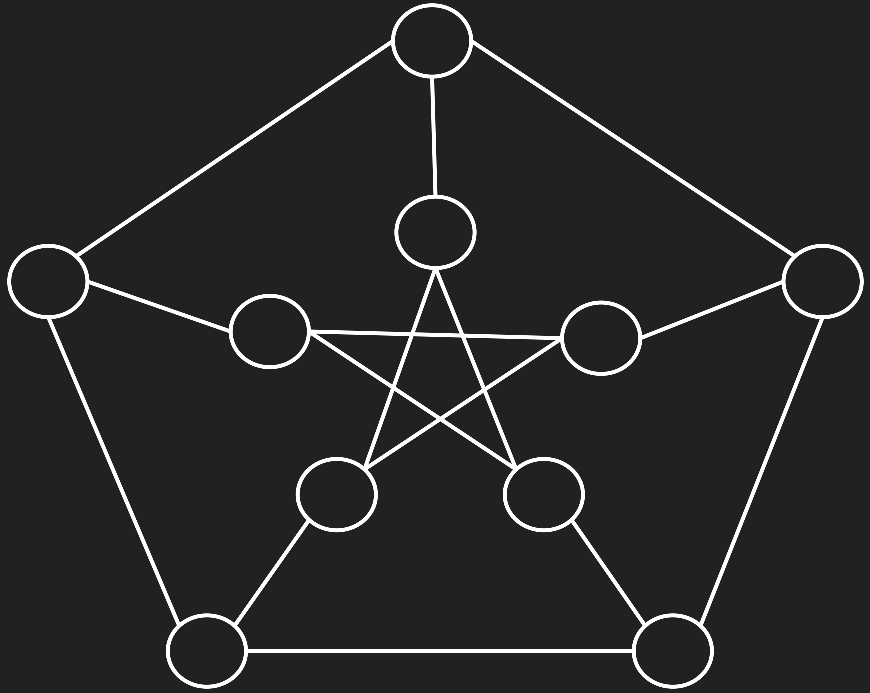
More Graph Problems / Complexity Theory

More Graph Problems

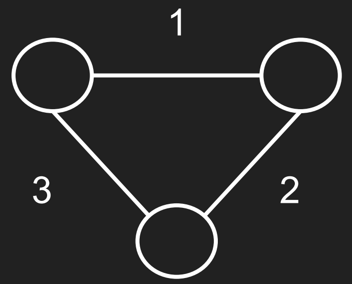


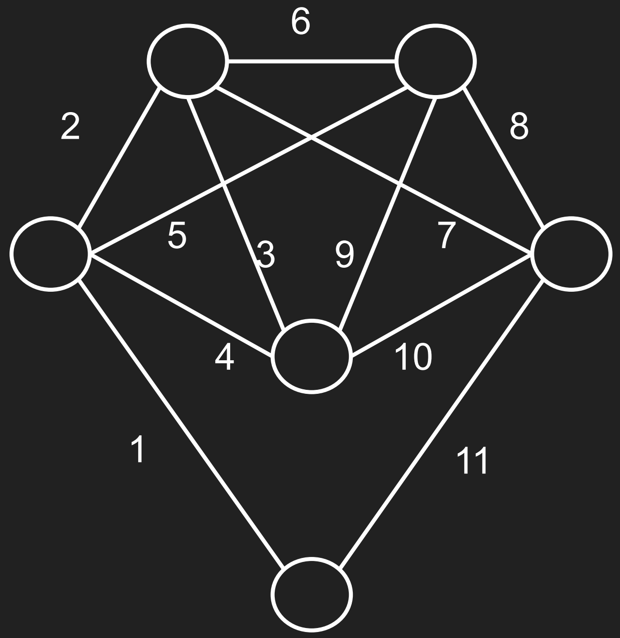
Petersen Graph

- Can you trace a path through this graph that travels along each edge exactly once?
- Can you trace a path through this graph that visits each vertex exactly once?

Eulerian Paths

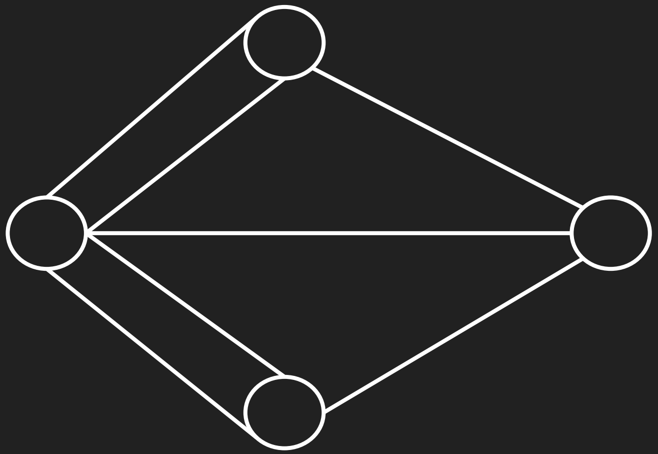
- Given a graph *G*, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) that visits each edge exactly once?





Eulerian Paths

- Given a graph *G*, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) that visits each edge exactly once?

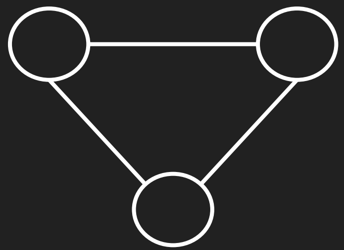


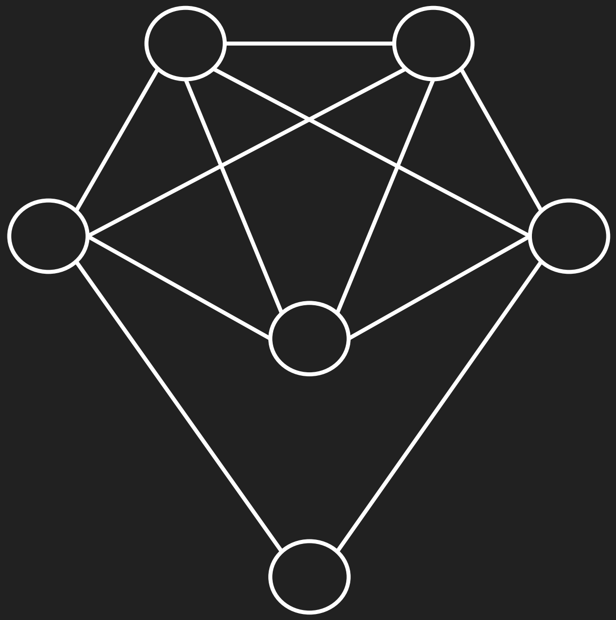
NO SOLUTION!

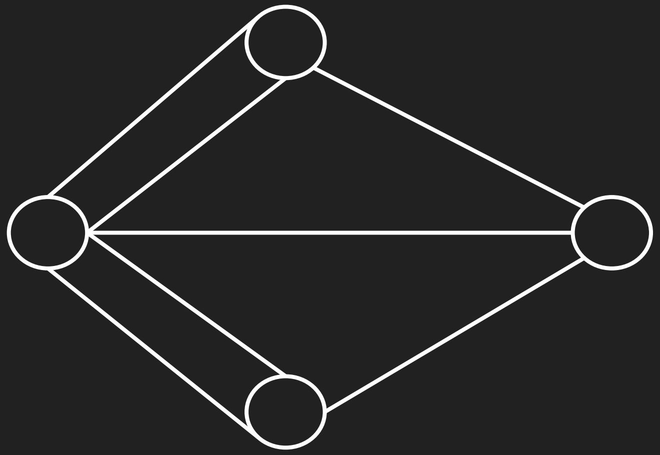
Seven Bridges of Königsberg

Eulerian Paths

- Key fact: there is an Eulerian circuit (or cycle) if and only if all vertices have even degree.
  - Intuition: If all vertices have an even degree, when you arrive at a vertex you must also have a way out.

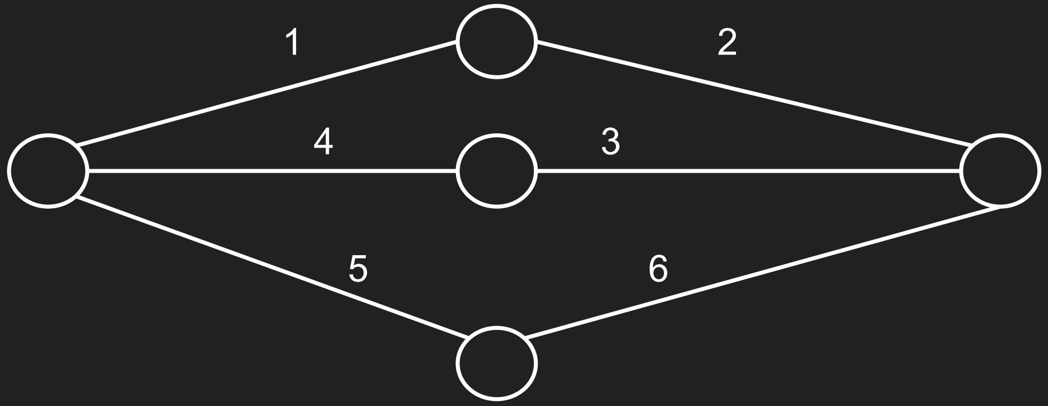






Eulerian Paths

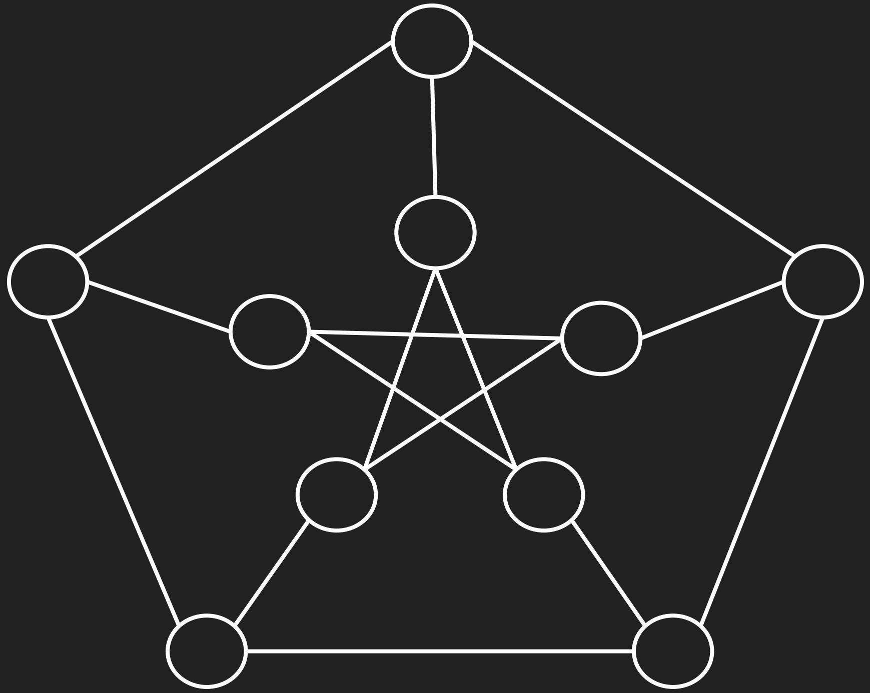
- Key fact: there is an Eulerian circuit (or cycle) if and only if all vertices have even degree.
  - Intuition: If all vertices have an even degree, when you arrive at a vertex you must also have a way out.
- There will still be an Eulerian path if exactly 2 of the vertices have odd degree



Eulerian Paths

- Key fact: there is an Eulerian circuit (or cycle) if and only if all vertices have even degree.
  - Intuition: If all vertices have an even degree, when you arrive at a vertex you must also have a way out.
- There will still be an Eulerian path if exactly 2 of the vertices have odd degree
- Algorithm:
  - Start at any vertex v, and start walking a path until you return to v (keeping track of which edges have been used).
  - If any vertex u along your path still has unused edges, go back to step 1 and find a circuit starting at u, then add that new path in the middle of the first path where u was reached.
  - Continue until no more vertices have unused edges
  - Runtime: $O(|E|)$

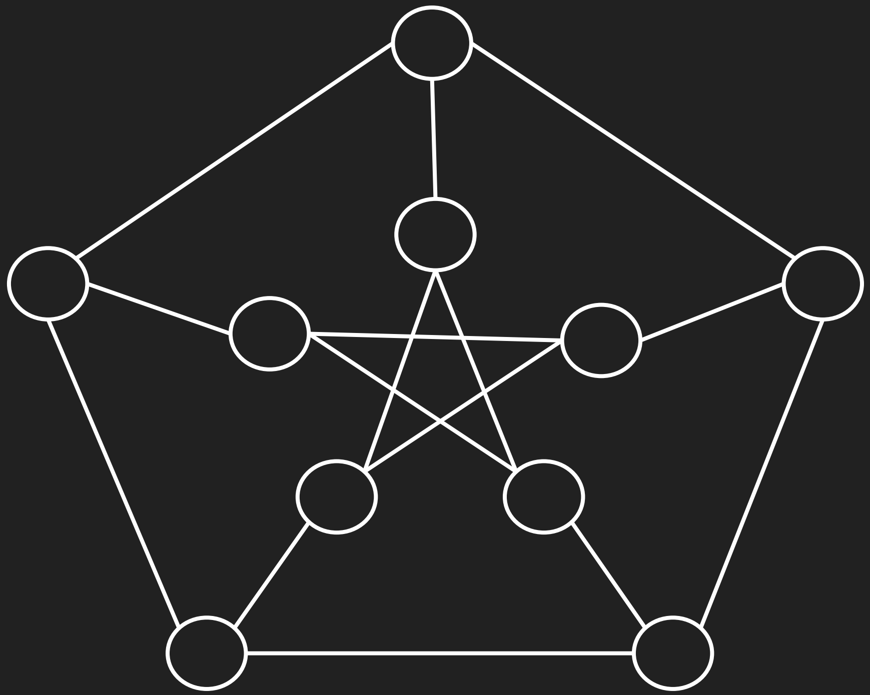
More Graph Problems



Petersen Graph

- Can you trace a path through this graph that travels along each edge exactly once?
- Can you trace a path through this graph that visits each vertex exactly once?

More Graph Problems

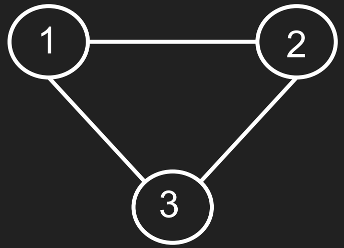


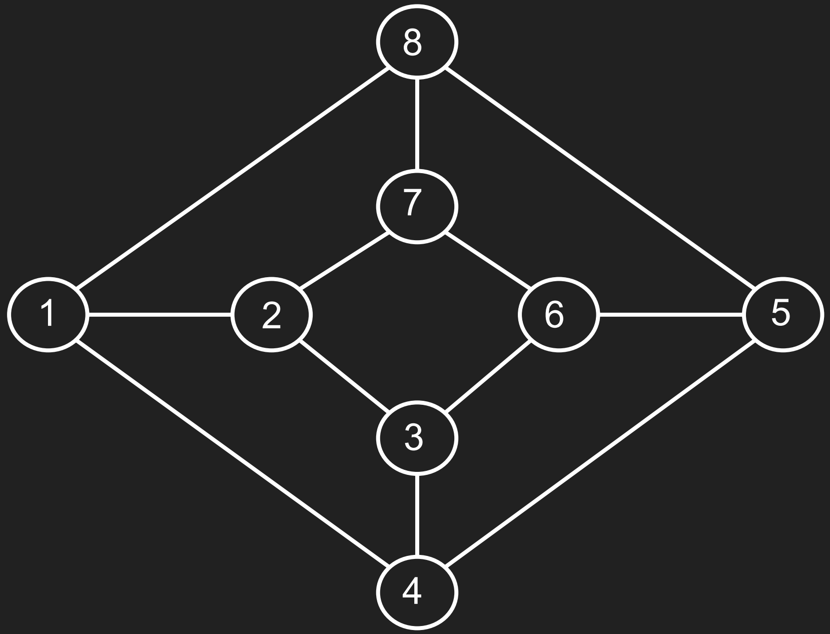
Petersen Graph

- ~~Can you trace a path through this graph that travels along each edge exactly once?~~ Polynomial Time Algorithm
- Can you trace a path through this graph that visits each vertex exactly once?

Hamiltonian Paths

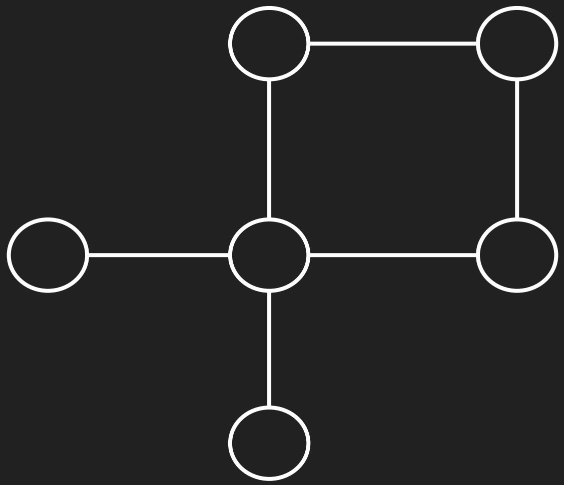
- Given a graph *G*, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) that visits each vertex exactly once?





Hamiltonian Paths

- Given a graph *G*, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) that visits each vertex exactly once?



NO SOLUTION!

Hamiltonian Paths

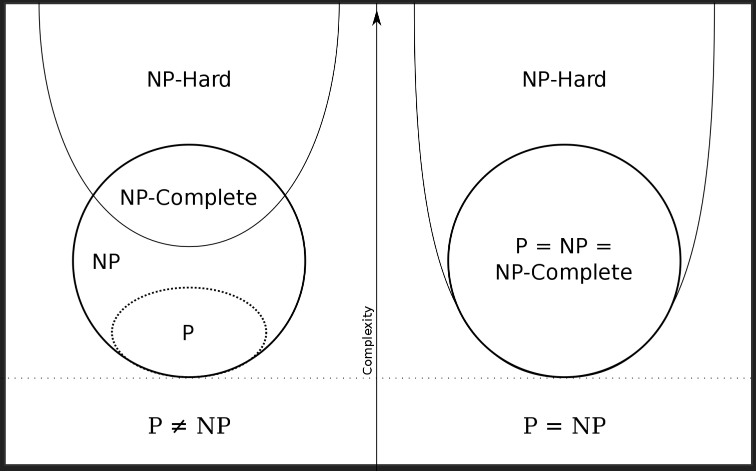
- Given a graph *G*, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) that visits each vertex exactly once?
- It turns out… we don't know how to find these paths quickly.
- Hamiltonian Paths is in a class of problems called NP-complete.

Complexity Theory

- Computational Complexity Theory is about grouping problems by their “difficulty.”
- *P* is the class of all yes-no problems than can be solved in polynomial time.
  - *FP* is the “functional equivalent” class, i.e. it’s not just yes-no problems
  - Most of the polynomial time algorithms we’ve discussed have a related yes-no version that is in *P*.
- *NP* is the class of all yes-no problems for which a given solution can be verified in polynomial time.
  - Everything in *P* is in *NP*.
- Important open problem in CS: does *P=NP?*
  - Most people think the answer is no

Complexity Theory

- *NP*-hard is the class of all problems that *NP* problems can be reduced to in polynomial time.
  - i.e. if you had a solution to one of the *NP*-hard problems, you would effectively have a solution to the *NP* problems.
- *NP*-complete problems are *NP*-hard problems that are also in *NP*.



Source: Wikipedia

Complexity Theory

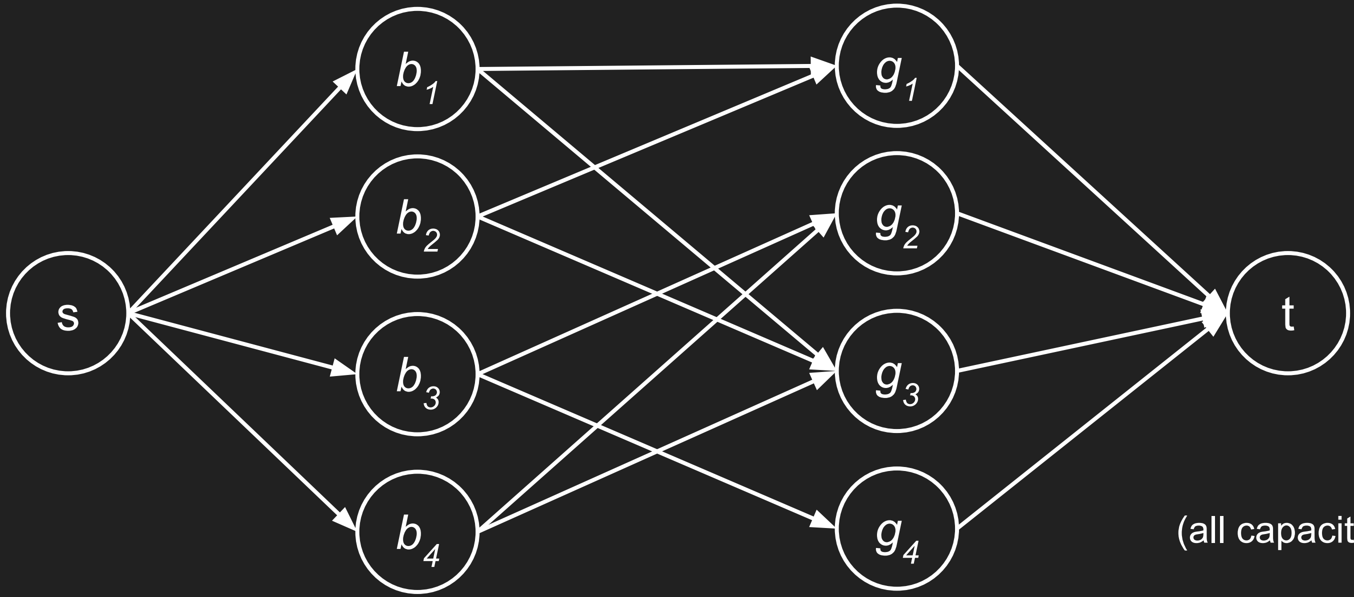
- *NP*-hard does NOT mean unsolvable!
- It just means we don't know whether it's possible to solve in polynomial time.
- Many *NP*-hard problems have reasonable polynomial approximation algorithms.
    - May or may not do a lecture on approximation algorithms later in the semester.
- Some *NP*-hard problems even have reasonable pseudo-polynomial algorithms.
    - Recall the Knapsack problem - it has a dynamic programming algorithm for finding an exact solution in $O(Wn)$.
- Generally if you need to solve an *NP*-hard problem today, you're either going to need to sacrifice time or accuracy.

Reductions

- Transforming an instance of problem *A* into an instance of problem *B* in a way that allows the solution to *B* to inform the solution of *A*.
  - Or, more usefully, determining a general algorithm for doing so for arbitrary instances of problem *A*.
- Not just a concept in complexity theory - we've actually already seen examples!
  - Recall the motivating examples for the max flow lecture: network connectivity, school dance, project management
  - Each of these were solved by reducing them to a max flow problem

Max Flow: School Dance

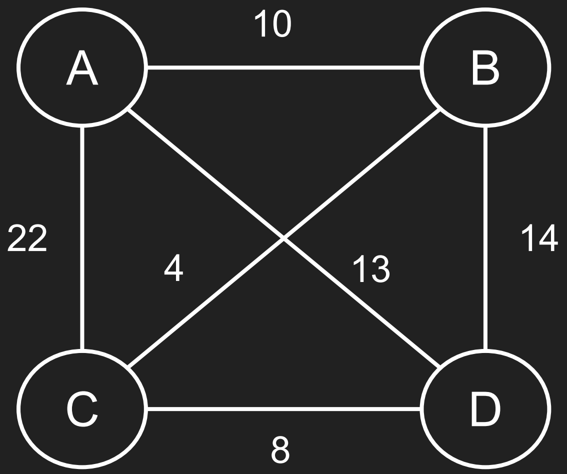
- Boys and girls need to be paired up for the school dance, but the kids only want to be paired with someone that they know. Is such a pairing possible? And if so, what's the pairing?



(all capacities = 1)

Traveling Salesman Problem

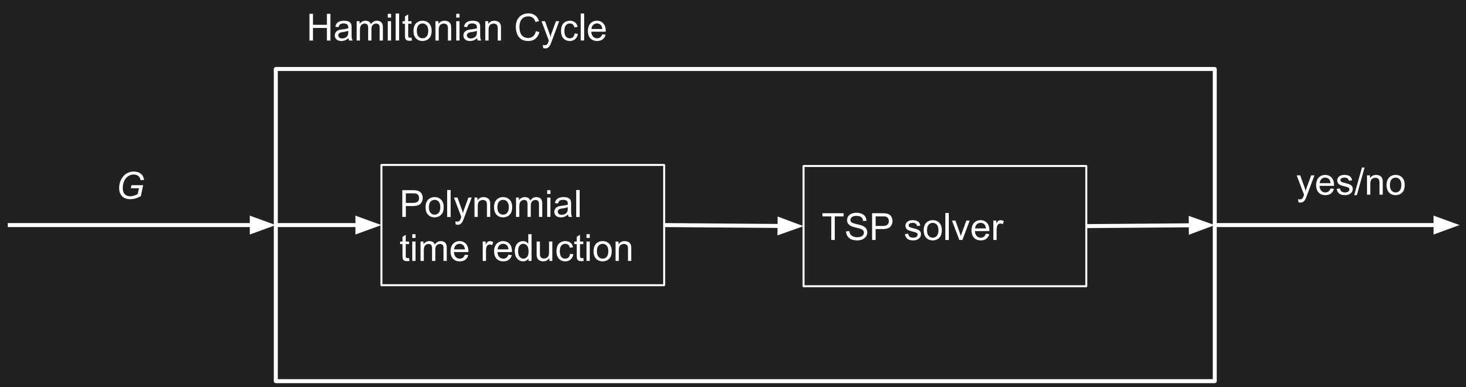
- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?
  - The decision version is whether there exists a route whose total distance is no greater than *L*.



Traveling Salesman Problem

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?
  - The decision version is whether there exists a route whose total distance is no greater than *L*.
- Seems awfully similar to Hamiltonian cycle
  - They’re both graph problems
  - They’re both looking for cycles
  - Biggest difference seems to be the edge weights
- Idea: we know Hamiltonian cycle is NP-complete. If we can reduce Hamiltonian cycle to Traveling Salesman Problem, we’ll know TSP is NP-complete as well.
  - Intuitively: we know Hamiltonian cycle is hard. If it’s easy to take the solution to TSP and determine a solution for Hamiltonian, that must mean TSP is hard too.

Traveling Salesman Problem



What's the polynomial time reduction?

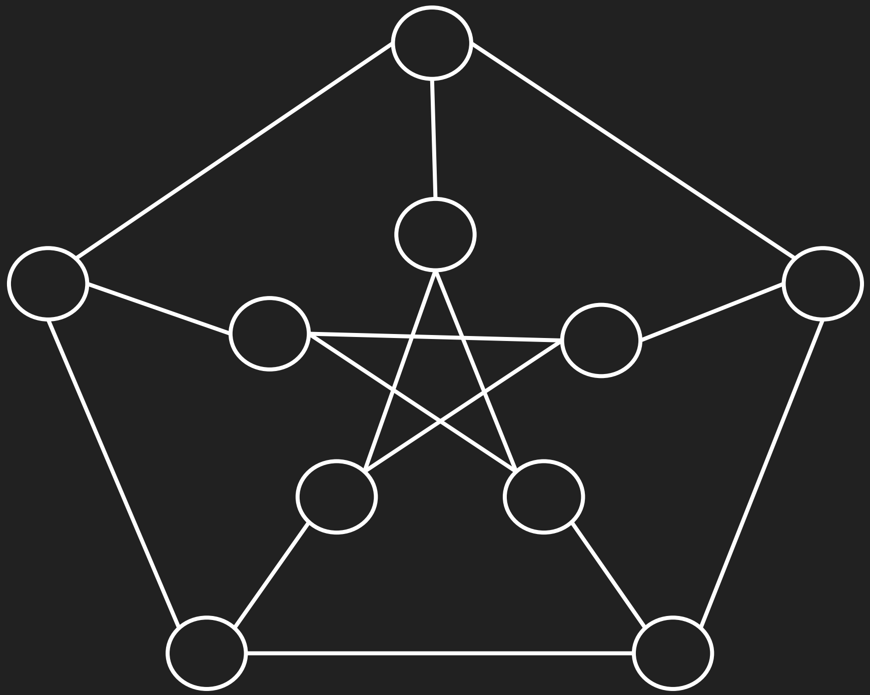
Traveling Salesman Problem

- Given graph $G = (V, E)$ for the Hamiltonian Cycle problem.
- Create a new graph $G' = (V, V \times V)$ where $e(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, otherwise $e(v_i, v_j) = 2$.
- Input G' into the Traveling Salesman Solver, where we're deciding whether a route exists with total distance $\leq |V|$.
- The solution to this TSP will be exactly the solution to whether the original graph G contains a hamiltonian cycle.
  - If TSP outputs yes for G', that means it found a path visiting all nodes while only using edges in E (otherwise it would have to use an edge with value 2 or visit a node more than once).
  - If a Hamiltonian cycle exists in G, then that cycle will also exist in G' and will have total distance equal to $|V|$, which would have TSP output yes.

Traveling Salesman Problem

- The reduction only involved creating *G'*, which can be done in polynomial time.
  - Therefore, the "hard part" must be in the TSP solver.
- TSP is *NP*-hard.
- TSP is in NP; given a route it's easy to check that it is valid and has total distance ≤ *L*.
- TSP is *NP*-complete.
- Note that this proof relies on the assumption that Hamiltonian Cycle is *NP*-complete.
  - This is true, but we haven't proven it here.

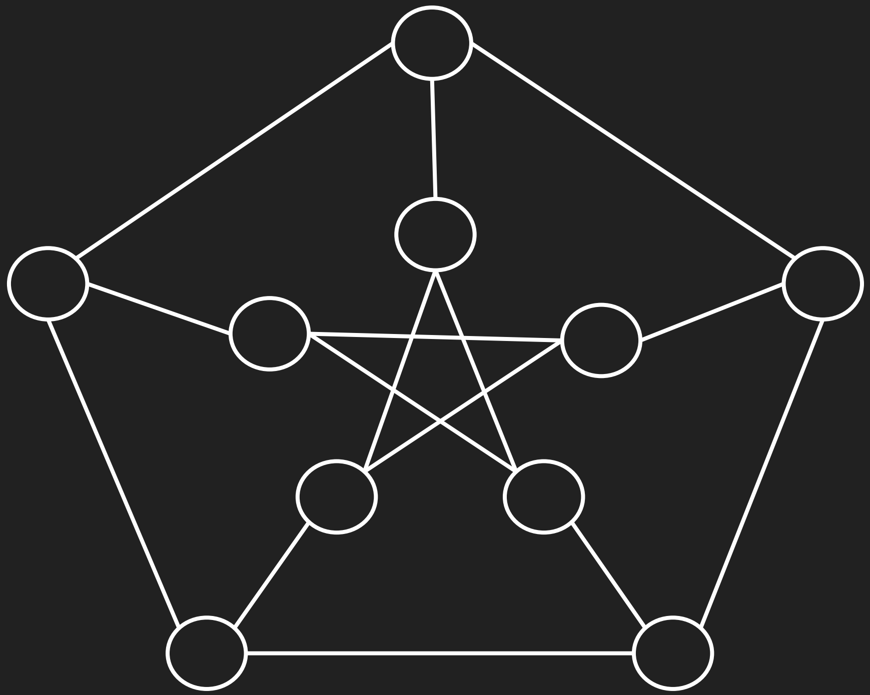
More Graph Problems



Petersen Graph

- ~~Can you trace a path through this graph that travels along each edge exactly once?~~ Polynomial Time Algorithm
- Can you trace a path through this graph that visits each vertex exactly once?

More Graph Problems



Petersen Graph

- ~~Can you trace a path through this graph that travels along each edge exactly once?~~ Polynomial Time Algorithm
- ~~Can you trace a path through this graph that visits each vertex exactly once?~~ NP-complete