We’ll return to NP-completeness now and look at some more classic hard problems. These problems are graph-based and so require “gadget reductions,” much like our previous hardness proof for INDEPENDENT-SET.

1 TSP and Hamiltonian Cycle

- Let’s discuss the Traveling Salesman Problem (TSP).
- You are given a graph $G$ (in general, directed) with $n$ vertices and non-negative edge weights.
- You want to find a path through $G$ that visits every vertex exactly once and returns to its starting point – a so-called “TSP tour” of $G$.

- In the optimization version of the problem, you seek a TSP tour such that the total weight of its edges is minimized.
- Conceptually, you’re a salesman that has to visit every city in your territory before returning home, and you know the cost of traveling between any two cities. You want to pay as little travel cost as possible.
- The decision version is, “does there exist a TSP tour with total weight at most $c$?”
- In the special case that every edge has weight 1, every TSP tour has weight exactly $n$, so the problem is just to find one such tour if it exists.
- Such a path is called a Hamiltonian cycle. The question of whether such a cycle exists in a graph $G$ is the HAMCYCLE problem.
We’ll study the hardness of HAMCYCLE, since if it is hard, so too is the more general TSP problem.

So how hard is HAMCYCLE?

- **Thm**: HAMCYCLE is NP-complete.
- **Pf**: first, show that HAMCYCLE is in NP.
  - Well sure – if there exists a cycle that touches every vertex exactly once, then the cycle itself can be the certificate, and verification is trivial.
  - For NP-hardness, we will reduce from 3SAT, i.e. show that 3SAT ≤ₚ HAMCYCLE.
  - Let φ be a 3CNF formula over n variables with m clauses.
  - WLOG, no clause of φ contains both a literal and its negation (as was the case for our SUBSET-SUM reduction.)
  - We’ll construct a graph G(φ) with 3mn + m + 2 vertices.
  - For each distinct variable xᵢ in φ, we will create a chain of 3m vertices – three per clause.
  - Label these vertices v^i_{j1}, v^i_{j2}, and v^i_{j3}.
  - The chain is connected bidirectionally: there are edges in both directions between
    - v^i_{j1} and v^i_{j2}
    - v^i_{j2} and v^i_{j3}
    - v^i_{j3} and v^i_{j+1,1}
  - Moreover, we will connect the chains for xᵢ and xᵢ₊₁ with edges between
    - v^i_{11} and v^{i+1}_{11}
    - v^i_{11} and v^{i+1}_{m3}
    - v^i_{m3} and v^{i+1}_{11}
    - v^i_{m3} and v^{i+1}_{m3}
  - We add vertices s and t, connect s to both ends of the chain for x₁, and connect both ends of the chain for xₙ to t. Add a back edge from t to s.
• Last, we create \( m \) clause vertices \( w_j \).

• If the literal \( x_i \) occurs in clause \( C_j \), we add two edges:
  
  \[
  v_{j2}^i \rightarrow w_j \\
  w_j \rightarrow v_{j3}^i
  \]

• If the literal \( \neg x_i \) occurs in clause \( C_j \), we add two edges:
  
  \[
  v_{j2}^i \rightarrow w_j \\
  w_j \rightarrow v_{j1}^i
  \]

• The resulting graph is our HAMCYCLE instance.

• It has polynomially many vertices and is trivially constructible from \( \phi \).

Whoa. Time for our iff.

• **Claim 1**: if \( \phi \) is satisfiable, \( G \) has a Hamiltonian cycle.

  • Let \( A \) be a satisfying assignment for \( \phi \).

  • Our cycle will traverse the chains from top to bottom (descending from \( s \) to \( t \)).

  • If \( A \) makes \( x_i \) true, traverse the chain for \( x_i \) from left to right (\( v_{11}^i \) to \( v_{m3}^i \)).

  • If \( A \) makes \( x_i \) false, traverse the chain for \( x_i \) from right to left (\( v_{m3}^i \) to \( v_{11}^i \)).

  • So far, this tour touches \( s \), \( t \), and each vertex in each chain once, but does not touch the \( w_j \)'s.

  • Now, for each clause \( C_j \), \( A \) makes at least one literal true.

  • Pick a true literal \( \ell_j \) for each \( C_j \).

  • If \( \ell_j = x_i \), then when we traverse the chain for \( x_i \), we “detour” from \( v_{j2}^i \) to \( w_j \) and then return to \( v_{j3}^i \).

  • If \( \ell_j = \neg x_i \), then when we traverse the chain for \( x_i \), we detour from \( v_{j2}^i \) to \( w_j \) and then return to \( v_{j1}^i \).

  • We detour exactly once for each \( w_j \), so the cycle touches each \( w_j \) once.

  • Conclude that this is a Hamiltonian cycle.

That was the easy direction.

• **Claim 2**: if \( G \) has a Hamiltonian cycle, \( \phi \) is satisfiable.

  • Let’s think about why this isn’t trivial.

  • Ideally, the cycle in \( G \) looks just like the one we designed for Claim 1 – it traverses each chain either left or right, with detours to touch each \( w_j \).
If so, we can read off a truth assignment from $G$: if chain $i$ is traversed left-to-right, set $x_i$ true; else set $x_i$ false.

We know that each $w_j$ is touched by exactly one detour, so by construction of $G$, some literal makes each clause true, and the resulting assignment is satisfying.

**What’s wrong with this argument?**

- We’re assuming a very specific form for the cycle! How do we know that this is the only form possible?
- If we ignore detours for a minute, it’s pretty clear that our cycle must traverse all of each chain before moving on to the next one.
- But, why can’t we detour from $v_{ij2}$ to $w_j$ and then move to $v_{ij3}^p$ or $v_{ij1}^p$ for some $p \neq i$?

We have to prove that our cycle has the right form.

**Pf:** first, we claim that, ignoring the detours, our cycle must traverse each chain in exactly one direction, from top to bottom beginning from $s$.

- Once we start traversing the chain for $x_i$, we can’t turn back without revisiting a vertex.
- Moreover, if we start traversing chains going down from $s$, we can’t “skip” a chain and go back up to it later, because we’ve touched one endpoint of the chain and so will get “stuck” repeating a vertex trying to get out of it.
- Now, let’s consider detours as well.
- Suppose we detour going left-to-right from $v_{ij2}$ to $w_j$ but do not immediately return to $v_{ij3}$.
- We eventually have to touch $v_{ij3}$, and we must do so coming from the right, since we cannot reuse $w_j$ or $v_{ij2}$.
- But then we get stuck, because we cannot get out of $v_{ij3}$.

- Hence, we must immediately return from our detour to the same chain.

- A symmetric argument shows that if we detour going right-to-left, we must immediately return to the same chain afterwards.

- Conclude that the cycle has the desired form, and we can read off the satisfying assignment as described. QED
2 3-Coloring a Graph

Let’s move on to another famous NP-complete problem.

- Let $G$ be an undirected graph.
- Is it possible to assign each vertex of $G$ one of three colors, such that no two endpoints of any edge are the same color?
- Not true for every graph. (Consider the 4-clique)
- Can generalize this problem to $k$-coloring for any $k > 1$.
- If $k = 2$, the problem is in $P$ (exactly the bipartite graphs can be 2-colored).
- **Claim**: 3-colorability for graphs is NP-complete.
- Pretty clearly in NP by the usual argument.
- We will show another reduction from 3-SAT.

**Claim**: 3-SAT $\leq_p$ 3-COLORABILITY.

- Let $\phi$ be a 3-CNF formula with variables $x_1 \ldots x_n$ and clauses $C_1 \ldots C_m$.
- We will build up a graph $G(\phi)$ in steps as follows.
- First, construct a triangle of vertices.

- Clearly, this triangle is 3-colorable.
- Designate its vertices arbitrarily as $t$, $f$, and $z$.
- Next, create $n$ more triangles by creating pairs of vertices $x_i$, $\bar{x}_i$. There are edges between $x_i$ and $\bar{x}_i$ and between these two vertices and $z$.

- Any valid 3-coloring of $G$ must assign $x_i$ and $\bar{x}_i$ distinct colors, neither of which is the color of $z$. 
• Hence, one will be colored the same as \( t \) and one the same as \( f \).
• (This is how we ensure that colorings of \( G \) will correspond to valid truth assignments of \( \phi \).)

But how do we encode satisfiability?

• For each clause \( C_j \), construct a widget \( W_j \) as follows.

• Connect each of the three singly-connected vertices of \( W_j \) to vertex \( t \) of the core triangle.
• Finally, connect these three vertices of \( W_j \) to vertices corresponding to the three literals of \( C_j \).
• \textit{Intuition:} \( W_j \) constrains the assignment implied by the coloring of the literals so that a 3-coloring of \( G \) cannot color all literals in a clause “false”.

Example for a simple formula:

• Onto the iff proof!
• \textbf{Claim 1}: if \( \phi \) is satisfiable, \( G \) is 3-colorable.
• Call our three colors “true”, “false”, and “green”.
• First, color \( t \) true, \( f \) false, and \( z \) green.
• Let \( A \) be a satisfying assignment to \( \phi \). Color each true literal vertex (according to \( A \)) true, and color its opposite false.
• So far, so good – no possible conflicts.

• Now, for each $W_j$, the three singly-connected vertices must be either false or green.

• Pick a vertex $v_j^*$ of $W_j$ connected to a true literal vertex (at least one must exist for $C_j$), and color it false. Color the other two vertices green.

• Finally, color the triangle of $W_j$ so that a green vertex is opposite $v_j^*$, and the other two vertices are true or false arbitrarily.

OK, but what about the other direction?

• **Claim 2**: if $G$ is 3-colorable, $\phi$ is satisfiable.

• Let $G$ be 3-colored, and name the colors of $t$, $f$, and $z$ “true”, “false”, and “green”.

• Exactly one vertex of $x_i$ and $\bar{x}_i$ must be colored true. Let $A$ be a truth assignment to the variables of $\phi$ makes the true-colored literals true.

• We claim that the assignment $A$ satisfies $\phi$. In particular, consider clause $C_j$.

• The three singly-connected vertices of $W_j$ are, as we established, either false or green. Moreover, at least one such vertex $v_j^*$ must be false, since it is opposite the green vertex of $W_j$’s triangle.

• Conclude that the literal vertex opposite $v_j^*$ must be colored true.

• Hence, each clause $C_j$ contains at least one true literal.