EECS 336: Algorithms **Homework 8 Solutions**

1 Secret Santa Cycles

We give an algorithm to solve this problem by reducing to bipartite matching. For each vertex in V, construct a vertex v_{out} on the left side and a vertex v_{in} on the right. For every edge $(u, v) \in E$, create an edge between u_{out} and v_{in}. Run a the algorithm for bipartite matching on this graph, and return true if and only if there is a perfect matching.

Correctness: Assume there is a perfect matching in the graph we construct. We build our collection of cycles as follows: for every pair (u, v) such that u_{out} is matched to v_{in}, include the edge (u, v). Because our matching is perfect, every vertex will have in degree one and out degree one. Because there are no self-edges, we get from basic graph theory that the graph is a collection of disjoint cycles containing every vertex.

Conversely, assume there is a collection of disjoint cycles containing every vertex. Then every vertex u has exactly one out edge, (u, v), for some v. Include $(u_{\text{out}}, v_{\text{in}})$ in the matching. Because every vertex has exactly one out edge, every vertex on the left will be matched, and because every vertex has exactly one in edge, every vertex on the right wil be matched.

Runtime: This algorithm constructs a matching graph with $O(n)$ vertices and $O(m)$ edges. The runtime is the time it takes to compute a maximum matching in a graph with $O(n)$ and $O(m)$ vertices.

2 Movie Seating

The reduction we use will be the composition of three reductions: we first reduce from directed Hamiltonian cycle to undirected Hamiltonian cycle. We will do this below. We will then reduce from undirected Hamiltonian cycle to Hamiltonian Path. This reduction is identical to the Hamiltonian cycle to Hamiltonian path reduction in the book (minus directedness of edges), and so we won't repeat it here. Then, we reduce from undirected Hamiltonian path to the movie seating problem. We do this reduction second.

2.1 Directed to Undirected Hamiltonian Cycle

Let $G = (V, E)$ be an instance of directed Hamiltonian Cycle. For each $v \in V$, construct three vertices: v_{in}, v_{mid}, and v_{out}, and and edges between v_{in} and v_{mid} and v_{mid} and v_{out}. For each edge (u, v) in E, construct an undirected edge between v_{out} and v_{in}. Now run the black box for undirected Hamiltonian Cycle on this instance, and return the result.

Correctness: Assume there is a directed Hamiltonian Cycle $P = v_1 v_2 \dots v_n$ in G. An undirected Hamiltonian Cycle is $v_{1\text{in}} v_{1\text{mid}} v_{1\text{out}} v_{2\text{in}} v_{2\text{mid}} v_{2\text{out}} \dots v_{n\text{in}} v_{n\text{mid}} v_{n\text{out}}$. Since the original Hamiltonian Cycle included all the vertices, so does the one we constructed. Since there is an edge between each

sucessive pair of vertices in P, there will be an edge between $v_{i\text{out}}$ and $v_{i+1\text{in}}$ for each i. The other edges are present by construction.

Now assume there is an undirected Hamiltonian Cycle in the constructed graph. Note that such a cycle (or its reverse) must visit vertices in the order $v_{i\text{in}} \to v_{i\text{mid}} \to v_{i\text{out}}$, for each of the n vertices. Now traverse the vertices G in the order that the three-vertex gadgets are traversed in the undirected Hamiltonian Cycle (or, again, its reverse, if need be). This yields an ordering of all the vertices (because every vertex in the constructed graph is included), and every successive pair is connected with an edge (since every edge $\{u_{\text{out}}, v_{\text{in}}\}$ corresponded to an edge in G.

Runtime: The runtime of this reduction is polynomial: it constructs a graph with $O(n)$ vertices and $O(m)$ edges, and calls the black box once.

2.2 Undirected Hamiltonian Path to Movie Seating

The reduction is much like the reduction showing TSP to be NP-hard. As input, we take an instance of undirected Hamiltonian Path $G = (V, E)$. We will construct a movie seating instance with a person for each $v \in V$. For every edge (u, v) in G, have u and v be acquaintainces. Have all other pairs not know each other. Run the movie seating black box on this instance with $k = 2n-2$, and return the result.

Correctness: Assume there is a Hamiltonian Path $P = v_1 \dots v_n$ in G. We can construct a seating configuration with value $2n-2$ by seating the people in the Hamiltonian path in that order. Note that each successive pair (u, v) is acquaintainces because there is an edge between u and v. Hence, the the seating configuration gets 2 points for each pair in the list, of which there are $n-1$.

Conversely, assume there is a seating configuration of value $2n-2$ (no more valuable configuration is possible). List the people in the order they are seated, and use this as the Hamiltonian path. Since each person is seated, this is a list of all the vertices of the graph. If there was a successive pair in this list without an edge, then there would be a pair in the seating configuration who did not know one another, meaning the seating configuration could not have value $2n-2$. Hence, the resulting ordering of vertices in G is a Hamiltonian Path.

Runtime: The above algorithm constructs an instance of movie seating with n people, and calls the black box once. This is polynomial.

3 More Cat People

We reduce from 3-SAT.

Construction: Given an instance of 3-SAT, construct a university for each clause j, with three delegates, labeled ℓ_{1j}, ℓ_{2j} and ℓ_{3j} - one for each literal in clause j. For each conflicting pair of literals in the 3-SAT instance, create a conflict pair between the corresponding literals. Run the Cat People black box on this instance and return the result.

Construction: Assume there is a satisfying assignment for 3-SAT. Then for each clause j, there is a literal ℓ_j^* which evaluates to true. For each university, take the corresponding delegate. Since

each clause is satisfied, each university has exactly one delegate. Since this delegation was built from a satisfying assignment, it doesn't have any pairs of delegates who are fighting.

Conversely, assume there is a delegation with no fights. For each university j, there is a delegate ℓ_j^* who has been invited. Set the variables so that the literals corresponding to these delegates all evaluate to true. Set any unassigned variables to be true. Note that every clause is satisfied because each one has a literal which we have forced to be true. Also note that no variable has been set to both true and false, as doing so would involve inviting two delegates who are fighting, which we know our delegation does not do. Hence, there is a satisfying, valid truth assignment for our 3-SAT instance.

Runtime: We construct a delegate for each literal, and at most $O(n^2)$ conflict pairs, where n is the number of variables in the 3-SAT instance. We call our black box once. This is polynomial.