

CIS 315 S08 HW 8. (9 points, 1 point EC)

34.2-2 (2 points)

Prove that if G is an undirected bipartite graph with an odd number of vertices, then G is nonhamiltonian.

By definition, a hamiltonian cycle consists of exactly n edges. If n is odd, the length of a hamiltonian cycle in G must also be odd. Observe that bipartite graphs do not permit odd-length cycles. Thus, G is nonhamiltonian. \square

34.2-7 (2 points)

Show that the hamiltonian-path problem can be solved on DAGs in polynomial time.

We can use a modification of TOPOLOGICAL-SORT. Any DAG that admits a hamiltonian path has exactly one source (vertex with in-degree 0). So we merely need to find a source in G , remove it and its incident edges, add it to the hamiltonian path, and recurse on the resulting graph.

We will always find at least one source, since by definition, G is acyclic. If at any point in the algorithm we find more than one source, G does not admit a hamiltonian path. We have already seen how to find sources in polynomial time, so it should be clear that this algorithm runs in polynomial time. An efficient implementation will run in $O(m + n)$ time, as with TOPOLOGICAL-SORT.

34.2-11 (3 points)

Prove that G^3 is hamiltonian.

Base case: $n = 3$. Trivial.

Inductive step: Assume all G^3 with $n \geq 3$ nodes are hamiltonian. Show that all G^3 with $n + 1$ nodes are hamiltonian.

Consider a graph G with $n + 1$ vertices and a spanning tree T of G . Observe that if T^3 is hamiltonian, certainly G^3 is hamiltonian. Let l be a leaf vertex in T with $\text{deg}(l) = 1$ and such that $(l, x) \in T$. Let T' be the graph formed by removing l and (l, x) . T' has n vertices and is connected, so by the inductive hypothesis $(T')^3$ is hamiltonian, which means that there exists a path $v_1, v_2, \dots, x, y, \dots, v_n, v_1$ in $(T')^3$.

Since $(l, x), (x, y) \in T$, $v_1, v_2, \dots, x, l, y, \dots, v_n, v_1$ is a hamiltonian path in $T^3 \Rightarrow T^3$ is hamiltonian $\Rightarrow G^3$ is hamiltonian. \square

34.4-2 (1 point EC)

Show the 3-CNF formula that results when we use the method of Thm 34.10 on formula (34.3).

:)

34.4-6 (2 points)

Suppose that we have an algorithm to decide formula satisfiability with running time $f(n)$. We can use this algorithm to compute a satisfying assignment using the *self-reducibility* of SAT.

For a formula ϕ with variables x_1, \dots, x_n , a satisfying assignment is just truth values for the variables in ϕ . So, the first step is to ask our SAT decider if ϕ is satisfiable. If not, we just return “no”.

If ϕ is satisfiable, then (for each level i of the algorithm) we must consider the formulas $\phi[x_i = 0]$ and $\phi[x_i = 1]$. At least one of these must be satisfiable if ϕ is satisfiable (if they are both satisfiable, we can just pick either one). We then recurse with the satisfying truth value of x_i substituted in ϕ .

We only need n calls to our SAT decider, since if $\phi[x_i = 0]$ is unsatisfiable, we can just recurse on ϕ with $x_i = 1$. The running time will be $O(n \cdot f(n))$, which is polynomial in n since $f(n)$ is assumed to be polynomial in n .