CS/ECE 374 Sec A ✧ Spring 2023

Homework 11

Due Wednesday, April 28, 2023 at 10am

---

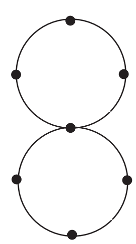
1. Recall that $L_u = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ is langauge of a UTM, and $L_{HALT} = \{\langle M\rangle \mid M \text{ halts on blank input}\}$ is the Halting language.

   - Let $L_{\text{regular}} = \{\langle M\rangle \mid M \text{ accepts a regular language}\}$.
     Prove that L_{regular} is undecidable.
   - Prove that $L_u \leq L_{HALT}$.
   - **Extra credit:** Prove that $L_{\text{emptylang}} = \{\langle M\rangle \mid L(M) = \emptyset\}$ is not recursively enumerable.

2. This problem is about polynomial time reductions and NP-Compleness.

   (a) SAT is a meta problem which partially explains why Cook-Levin proved that it is NP-Complete first. In this part the goal is to get some practice modeling problems via constraint satisfaction, in other words, reducing them to SAT. Given an undirected graph $G = (V, E)$ a *matching* in G is a set of edges $M \subseteq E$ such that no two edges in M share a node. A matching M is *perfect* if $2|M| = |V|$, in other words if every node is incident to some edge of M. PerfectMatching is the following decision problem: does a given graph G have a perfect matching? Describe a polynomial-time reduction from PerfectMatching to SAT. *Hint: use a Boolean variable x_e for each edge $e \in E$ and write appropriate constraints.* Does this prove that PerfectMatching is NP-Complete?

   (b) We call an undirected graph an *eight-graph* if it has an odd number of nodes, say $2n - 1$, and consists of two cycles C_1 and C_2 on n nodes each and C_1 and C_2 share exactly one node. See figure below for an eight-graph on 7 nodes.

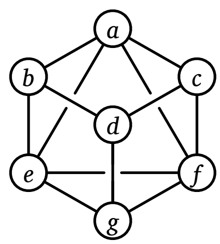
   Given an undirected graph G and an integer k, the EIGHT problem asks whether or not there exists a subgraph which is an eight-graph on $2k - 1$ nodes. Prove that EIGHT is NP-Complete.

3. **Not to submit:** Given an undirected graph $G = (V, E)$, a partition of V into $V_1, V_2, \ldots, V_k$ is said to be a clique cover of size k if each V_i is a clique in G. CLIQUE-COVER is the following decision problem: given G and integer k, does G have a clique cover of size at most k?

- Describe a polynomial-time reduction from CLIQUE-COVER to SAT. Does this prove that CLIQUE-COVER is NP-Complete? For this part you just need to describe the reduction clearly, no proof of correctness is necessary. *Hint:* Use variablex $x(u, i)$ to indicate that node u is in partition i.
- Prove that CLIQUE-COVER is NP-Complete.

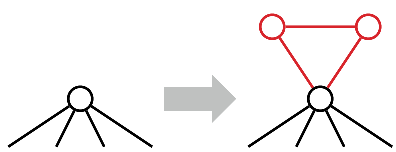
Solved Problem

4. A *double-Hamiltonian tour* in an undirected graph G is a closed walk that visits every vertex in G exactly twice. Prove that it is NP-hard to decide whether a given graph G has a double-Hamiltonian tour.



This graph contains the double-Hamiltonian tour $a \to b \to d \to g \to e \to b \to d \to c \to f \to a \to c \to f \to g \to e \to a$.

Solution: We prove the problem is NP-hard with a reduction from the standard Hamiltonian cycle problem. Let G be an arbitrary undirected graph. We construct a new graph H by attaching a small gadget to every vertex of G. Specifically, for each vertex v, we add two vertices $v^\sharp$ and $v^\flat$, along with three edges $vv^\flat$, $vv^\sharp$, and $v^\flat v^\sharp$.



A vertex in G, and the corresponding vertex gadget in H.

I claim that G has a Hamiltonian cycle if and only if H has a double-Hamiltonian tour.

$\Longrightarrow$ Suppose G has a Hamiltonian cycle $v_1 \to v_2 \to \cdots \to v_n \to v_1$. We can construct a double-Hamiltonian tour of H by replacing each vertex v_i with the following walk:

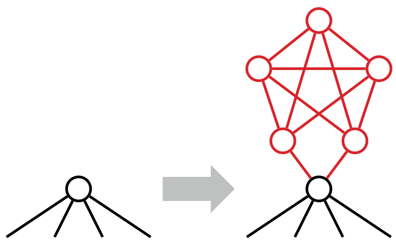
$$\cdots \to v_i \to v_i^\flat \to v_i^\sharp \to v_i^\flat \to v_i^\sharp \to v_i \to \cdots$$

$\Longleftarrow$ Conversely, suppose H has a double-Hamiltonian tour D. Consider any vertex v in the original graph G; the tour D must visit v exactly twice. Those two visits split D into two closed walks, each of which visits v exactly once. Any walk from $v^\flat$ or $v^\sharp$

to any other vertex in H must pass through v. Thus, one of the two closed walks visits only the vertices v, $v^\flat$, and $v^\sharp$. Thus, if we simply remove the vertices in $H \setminus G$ from D, we obtain a closed walk in G that visits every vertex in G once.

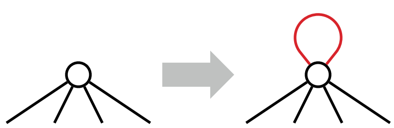
Given any graph G, we can clearly construct the corresponding graph H in polynomial time.

With more effort, we can construct a graph H that contains a double-Hamiltonian tour ***that traverses each edge of H at most once*** if and only if G contains a Hamiltonian cycle. For each vertex v in G we attach a more complex gadget containing five vertices and eleven edges, as shown on the next page. ■



A vertex in G, and the corresponding modified vertex gadget in H.

Common incorrect solution (self-loops): We attempt to prove the problem is NP-hard with a reduction from the Hamiltonian cycle problem. Let G be an arbitrary undirected graph. We construct a new graph H by attaching a self-loop every vertex of G. Given any graph G, we can clearly construct the corresponding graph H in polynomial time.

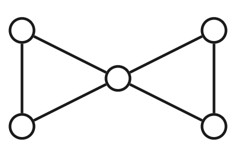


An incorrect vertex gadget.

Suppose G has a Hamiltonian cycle $v_1 \to v_2 \to \cdots \to v_n \to v_1$. We can construct a double-Hamiltonian tour of H by alternating between edges of the Hamiltonian cycle and self-loops:

$$v_1 \to v_1 \to v_2 \to v_2 \to v_3 \to \cdots \to v_n \to v_n \to v_1.$$

On the other hand, if H has a double-Hamiltonian tour, we *cannot* conclude that G has a Hamiltonian cycle, because we cannot guarantee that a double-Hamiltonian tour in H uses *any* self-loops. The graph G shown below is a counterexample; it has a double-Hamiltonian tour (even before adding self-loops) but no Hamiltonian cycle.



This graph has a double-Hamiltonian tour.

✧

Rubric (for all polynomial-time reductions): 10 points =

+ 3 points for the reduction itself
  - For an NP-hardness proof, the reduction must be from a known NP-hard problem. You can use any of the NP-hard problems listed in the lecture notes (except the one you are trying to prove NP-hard, of course).
+ 3 points for the "if" proof of correctness
+ 3 points for the "only if" proof of correctness
+ 1 point for writing "polynomial time"

- An incorrect polynomial-time reduction that still satisfies half of the correctness proof is worth at most 4/10.
- A reduction in the wrong direction is worth 0/10.