Homework 10

CS/ECE 374 B

Due 8 p.m. Tuesday, December 10

Question 1: From each clause according to its ability to be satisfied .... 10 points

A disjunctive normal form formula is the converse of CNF; i.e., it is an <u>or</u> of a number of clauses where each clause is an <u>and</u> of some terms. E.g.:

$$(x \wedge y \wedge z) \vee (z \wedge \overline{y} \wedge w) \vee (x \wedge \overline{z})$$

DNF-SAT is the analog problem of CNF-SAT: given a DNF formula f determine if there is a satisfying assignment of the corresponding variables that renders the formula true.

(4) (a) Design and analyze an <u>efficient</u> algorithm for DNF-SAT.

(4) (b) Demonstrate a reduction from 3SAT to DNF-SAT and analyze its runtime. (Hint: use the distributive law.)

(2) (c) Why does this not prove that $P = NP$?

Question 2: Solomonic decision problem .... 10 points

(3) (a) Suppose you are given an algorithm `partitionable`, given a set X of positive integers, determines whether X can be partitioned into two sets A and B such that $\sum A = \sum B$. (`partitionable` returns True or False). Design an analyze an "efficient" (see below) algorithm that computes such a partition if it exists. In other words, on an input X you should return two sets A and B such that $X = A \cup B$, $A \cap B = \emptyset$, and $\sum A = \sum B$, or return an error if such partition does not exist.

Your algorithm should call `partitionable` as a subroutine; its efficiency will therefore depend on the efficiency of `partitionable`. You should analyze both the amount of work that is done within your algorithm, and the number of calls to `partitionable` that are made, expressing both in asymptotic terms (e.g., "The algorithm performs $\Theta(N^2)$ work and makes $\Theta(N)$ calls to `partitionable`"). Your algorithm should run in polynomial time under the assumption that `partitionable` runs in polynomial time.

(3) (b) Design and analyze an efficient algorithm for 2PARTITION: given a set X of $2n$ integers, partition them into n disjoint pairs such that the sum of each pair is equal. I.e., given X with $|X| = 2n$, create sets $S_1, \ldots, S_n$ such that $|S_i| = 2$, $S_i \cap S_j = \emptyset$ for $i \neq j$, $X = \bigcup_{i=1}^{n} S_i$ and $\sum S_i = \sum S_j$ for any $1 \leq i, j \leq n$.

(4) (c) Show that the problem 7PARTITION is NP-Hard. 7PARTITION is defined as above, taking a set of $7n$ integers and splitting them into n disjoint sets of size 7 with the sum of each set being equal. You may assume that 3PARTITION is NP-hard.

Question 3: Charon's crossing .... 10 points

Solve problem 42 in Chapter 12 of the textbook. Below is a restatement:

You are given a graph $G = (V, E)$ and the number k. The problem can be described as defining a sequence of subsets of vertices $X_0, \ldots, X_{2m+1}$ with the following properties:

- $X_0 = V$, $X_{2m+1} = \emptyset$
- X_i and X_{i+1} differ by at most k vertices.
- X_{2i+1} is an independent set (in G) for all $i = 0, \ldots, m$
- $V - X_{2i}$ is an independent set (in G) for all $i = 0, \ldots, m$

Less formally, you can think about having two sets X and Y. All vertices start X and at each odd "move" you shift up to k vertices from X to Y, at each odd "move" you shift up to k vertices from Y to X. After every odd move, X cannot have two vertices that are connected by an edge; after every even move $Y (= V - X)$ cannot contain two vertices that are connected by an edge.

The decision problem CHARON is: given a graph G and a number k, is there a sequence of valid moves that ends with $X = \emptyset$? Your job is to show that CHARON is NP-hard.

As an example, for the classical formulation where $V = \{\text{Goat}, \text{Wolf}, \text{Cabbage}\}$, $E = \{\text{Goat} - \text{Cabbage}, \text{Wolf} - \text{Goat}\}$, and $k = 2$ we have a solution where:

$X_0 = \{\text{Goat}, \text{Wolf}, \text{Cabbage}\}$	initial configuration
$X_1 = \{\text{Wolf}, \text{Cabbage}\}$	move Goat
$X_2 = \{\text{Wolf}, \text{Cabbage}\}$	empty move
$X_3 = \{\text{Cabbage}\}$	move Wolf
$X_4 = \{\text{Goat}, \text{Cabbage}\}$	move Goat
$X_5 = \emptyset$	move Goat, Cabbage, final configuration

Question 4: Semiprime kind of life............10 points (bonus)

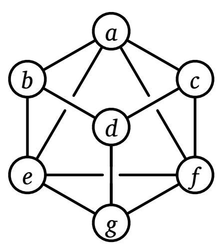
Find the prime factors of the following integer:

41202343698665954385553136533257594817981169984432798284545562643387644556
52484261980988704231618418792614202471888694925609317763750334211309823974
85150944909106910269861031862704114880866970564902903653658867433731720813
10410519086425479328260139125762403394637326939 1

Solved Problem

Question 5: Encore............

A <u>double-Hamiltonian tour</u> 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.



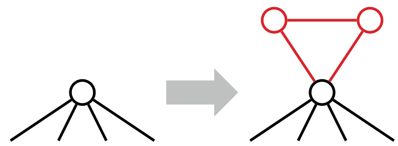
This graph contains the double-Hamiltonian tour $a \to b \to d \to g \to e \to b \to d \to c \to f \to a \to c \to f \to g \to e \to a$.

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 $v^\flat v^\sharp$.

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:

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

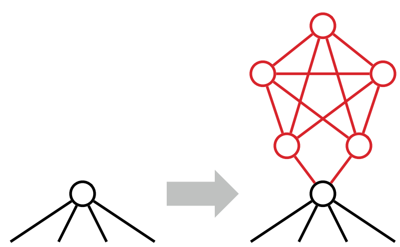


A vertex in G, and the corresponding vertex gadget in H.

$\Longleftarrow$ 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.

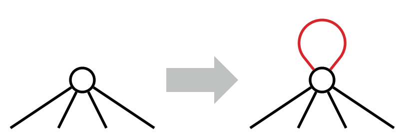
Given any graph G, we can clearly construct the corresponding graph H in polynomial time.

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.



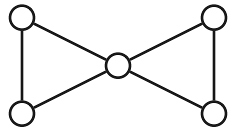
An incorrect vertex gadget.

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.