Tractability vs Intractability (Part II)

Remind that

- FPSPACE denotes the class of functions Polynomial Space computable,
- FP denotes the class of functions Polynomial Time computable, and
- co-NP $= \{A | \overline{A} \in \mathsf{NP}\}$

1. **Balanced Colors.** A k-coloring is balanced if exactly $1/k$ of the vertices have each color. Show that for any $k \geqslant 3$ the problem of whether a graph has a BALANCED k-COLORING is NP-complete. Then show that it is in P for $k = 2$.

2. **Really independent sets.** Given a graph $G = (V, E)$, say that a subset $S \subseteq V$ is *really independent* if there are no $u, v \in S$ that are two or fewer steps apart in the graph, that is, for all $u, v \in S$, the length of the shortest path between u and v is at least 3. The problem REALLY INDEPENDENT SET takes a graph G and an integer k as input, and asks if G has a really independent set of size k or more. Prove that this problem is NP-complete.

3. **Proving NP-completeness by generalization.** For each of the poblems below, prove that is NP-complete by showing that is a generalization of some classical NP-complete problems.

   (a) SUBGRAPH ISOMORPHISM: Given two undirected graphs G and H, is G a subgraph of H?

   (b) MAX SAT: Given a boolean formula F and an integer k, decide whether there is an assignment that satisfies at least k clauses.

   (c) DENSE SUBGRAPH: Given a graph G and two integers a, b, decide whether there exist a vertices of G such that there are at least b edges between them.

   (d) SPARSE SUBGRAPH: Given a graph G and two integers a, b, decide whether there exist a vertices of G such that there are at most b edges between them.

   (e) RELIABLE NETWORK: We are given to $n \times n$ matrices, a *distance* matrix d_{ij} and a *connectivity requirement* matrix r_{ij}, as well as a budget b; we must find a graph $G(\{1, \ldots, n\}, E)$ such that (1) the total cost of all edges is b or less and (2) between any two distinct vertices i and j there are r_{ij} vertex-disjoint paths.

4. **Feedback set.** Many algorithmic problems are easier to solve or are only well-defined for directed acyclic graphs. For instance, in task scheduling, the edges may represent precedence scheduling constraints, and we need to identify the smallest number of constraints that must be dropped so as to permit a valid schedule; namely, the graph of tasks and constraints does not contain directed cycles. Given a directed graph $G = (V, E)$, a subset E' of E is a *feedback arc set* if by deleting all the edges in E' from G, we obtain an acyclic graph.

   FAS: Given a directed graph $G = (V, E)$ and an integer k, does there exist a feedback arc set of size k?

   Show that FAS is NP-complete.

   (*Hint*: You might consider a reduction from VERTEX COVER. On input (G, k) where G is an undirected graph $G = (V, E)$ and we want to know if it contains a vertex cover of size k. We construct a directed graph $G' = (V', E')$ as follow. Let the vertices of G be $v_1, \ldots v_n$. Then, G' has $2n$ vertices $w_1, w'_1, \ldots, w_n, w'_n$ and edges (w_i, w'_i) for $i = 1, \ldots, n$ and $(w'_i, w_j), (w'_j, w_i)$ for every edge $(v_i, v_j) \in E$.)

5. **Node disjoint paths.** In the NODE DISJOINT PATHS problem the input is an undirected graph in which some vertices have been specially marked: a certain number of "sources" $s_1, s_2, \ldots, s_k$ and an equal number of "destinations" $t_1, t_2, \ldots, t_k$. The goal is to

find k node disjoint paths (that is, paths which have no nodes in common) where the ith path goes from s_i to t_i. Show that this problem is NP-complete.

Here is a sequence of progressively stronger hints.

(a) Reduce from 3SAT.

(b) For a 3SAT formula with m clauses and n variables, use $k = m + n$ sources and destinations. Introduce one source/destination pair (s_x, t_x) for each variable x, and one source/destination pair (s_C, t_C) for each clause C.

(c) For each clause introduce 6 new intermediate vertices, one for each literal occurring in that clause and one for its complement.

(d) Notice that if the path from s_C to t_C goes through some intermediate vertex representing, say, an occurrence of a variable x, then no other path can go to that vertex. What vertex would you like the other part to be forced to go through instead?

6. **Search vs decision.** Suppose you have a procedure which returns in polynomial time and tells you whether or not a graph has a *Hamiltonian path*. Show that you can use it to develop a polynomial time algorithm for HAMILTONIAN PATH (which return the actual path, if exists).

7. **Prime Factorization.** Although that factoring integers is an ancient and honored problem, and number theorists never needed extra motivation to work on it, the advent of public-Key cryptography (see chapter 12 [Papadimitriou 94]) gave to this classical problem an unexpected practical significance. Remind that by the fundamental theorem of arithmetic, every positive integer has a unique prime factorization. However, the fundamental theorem of arithmetic gives no insight into how to obtain an integer's prime factorization; it only guarantees its existence. Let us denote this computational problem as FACTORS.

   When discussing what complexity classes the integer factorization problem FACTORS falls into, it's necessary to distinguish two slightly different versions of the problem:

   FACTORIZATION (*A functional problem related to* FACTORS): Given an integer x, find an integer p with $1 < p < x$ that divides x (or conclude that x is prime).

   FACTORIZATION-D (*A decisional version of* FACTORIZATION): Given two integers x and y with $1 \leqslant y \leqslant x$ does x have a factor p with $1 < p < y$?

   One of the most significant results in Computational Complexity is that the problem of deciding whether a given integer is prime (PRIMES problem) can be solved in polynomial time (Manindra Agrawal, Neeraj Kayal, Nitin Saxena, "PRIMES is in P."Annals of Mathematics 160(2): 781-793 (2004)).

   (a) Show that FACTORIZATION-D is in both NP and co-NP. (It is suspected to be outside all the three complexity classes P, NP-complete or co-NP-complete. If it could be proved that is either NP-complete or co-NP-complete, that would imply NP = co-NP.)

   (b) Show that, if P = NP then FACTORIZATION $\in$ FP.

   (c) Show that, if P = NP then FACTORS $\in$ FP, i.e. computing the exact factorization of a given integer x can be done in polynomial time (If P = NP then the **RSA cryptosystem** can be broken in polynomial time!).

8. **Maximum clique.** Let us denote by MAX-CLIQUE the problem defined as follows:

   Given a graph $G = (V, E)$ compute a *clique of maximum size*. That is, compute $U \subseteq V$ such that $\forall u, v \in U$, $u \neq v$, then $(u, v) \in E$ and $\forall U'$, $U' \subseteq V$ with $|U'| > |U|$ then $\exists u', v' \in U'$ such that $(u', v') \notin E$.

   (a) Show that MAX-CLIQUE$\in$ FPSPACE.

(b) Show that $\mathsf{P} = \mathsf{NP}$ if and only if MAX-CLIQUE$\in \mathsf{FP}$.

9. **Succint formula.** Remind that we say two boolean formulas are *equivalent* if they have the same set of variables and are true on the same set of assignments to those variables (i.e., they describe the same boolean function). A boolean formula is *minimal* if no shorter Boolean formula is equivalent to it. Let MIN-FORMULA be the collection of minimal Boolean formulas. Show that if $\mathsf{P} = \mathsf{NP}$, then MIN-FORMULA $\in \mathsf{P}$.

10. **Max number of sat clauses.** Let us denote by MAX-SAT the problem defined as follows:

    Given a boolean formula Φ in Conjunctive Normal Form, compute a boolean assignment a that satisfies the maximum number of clauses. That is, if a satisfies k clauses of Φ, then there is no assignment a' satisfying more than k clauses.

    (a) Show that MAX-SAT $\in \mathsf{FPSPACE}$.

    (b) Show that $\mathsf{P} = \mathsf{NP}$ if and only if MAX-SAT$\in \mathsf{FP}$.

11. **Geography game.** One way to pass the time during long journey is to play the word game of *Geography*. The first player names a place, say Barcelona. The second player replies with a place whose name starts with the same letter the previous place ended with, such as Athens. The first player could then respond with Sicily, and so on. Each place can only be named once, and the first player who cannot make a move loses.

    We can make a mathematical version of this game. We have a directed graph $G = (V, E)$, and a designated start node $s \in V$. The nodes represent the places and the directed edges represent the legal moves. We start at s, and you and I take turns deciding which step to do next. Whoever ends up in a dead end, where every outgoing edge points to a vertex we have already visited, loses. A position or node v is a *winning position* for the first player if whatever legal move the second player selects, the first player never ends up in a dead end.

    GEOGRAPHY

    Input: An directed graph $G = (V, E)$ and a start node s.

    Question: Is s a *forced* win position for the first player? (The first player with strategy s always wins whatever is the strategy selected by the second player).

    Show that GEOGRAPHY is in PSPACE. (In fact GEOGRAPHY is a well known PSPACE-complete problem).

12. **Geography with repeat visits.** Suppose we play GEOGRAPHY without the restriction that each vertex we can only be visited at most once. However, the graph is still directed, and you lose if you find yourself at a vertex with no outgoing edges. Show that the problem of telling whether the first player has a winning strategy in this case is in P. (Notice that if a node is reached after a number of turns greater than $n - 1$, then, for sure there is a draw).

13. **Chess.** Consider a board game such as Go or Chess, where the state space of possible positions on an $n \times n$ board is exponentially large and there is no guarantee that the game only lasts for a polynomial number of moves. Assume for the moment that there is no restriction on visiting the same position twice.

    (i) Show that, given configuration of a $n \times n$ board, telling whether the first player of has a *forced* win strategy is in EXP (CHESS or GO are well known EXP-complete problems).

    (iii) Show that if there are a guarantee that the game only lasts a polynomial number of moves, then such a problem will be in PSPACE.

14. **Let's call it a draw.** Modern Chess has a "50 move rule" in which a player can claim a draw, if for 50 consecutive moves, no piece has been captured and no pawn has been moved. Suppose that we generalize this to $n \times n$ boards with a function $f(n)$, such that there is a draw if no captures or pawn moves have occurred in $f(n)$ consecutive moves. Show that if $f(n) = poly(n)$, this version of CHESS is in PSPACE.

15. **Unbounded boolean formula game**. Consider a game played on a given CNF Boolean formula F defined on a set of variables $X \cup Y$ and a given assignment α. Players I and II take turns. Player I (II) moves by changing at most one variable in X (Y); passing is allowed. Player I wins if F ever becomes true. In other words, Player II cannot move from α to α' if α satisfies F, but Player I can move from α to α' provided that α and α' differ only in the assignment to at most one variable in X. Note that there is no provision for Player II to ever win; the most he can hope to accomplish is a draw, by preventing Player I from ever winning.

    UBF GAME

    Input: A CNF Boolean formula F, sets of variables X and Y, and an assignment α.

    Question: Does Player I have a forced win?

    Show that UBF GAME is in EXP (in fact is EXP-complete [L. Stockmeyer, A.K. Chandra, 1979].