

1. A square matrix M is lower triangular if each entry above the main diagonal is zero, that is, each entry $M_{i,j}$, with $i < j$, is equal to zero. Show that if there is an $O(n^2)$ time algorithm for multiplying two n by n lower triangular matrices then there is an $O(n^2)$ time algorithm for multiplying two arbitrary n by n matrices.

†

2. Show that if there is an $O(n^k)$, $k \geq 2$, time algorithm for inverting a nonsingular n by n matrix C then there is an $O(n^k)$ time algorithm for multiply two arbitrary n by n matrices A and B .

For a square matrix A , A inverse, denoted A^{-1} , is the unique matrix such that $AA^{-1} = I$, where I is the identity matrix with 1's on the main diagonal and 0's everywhere else. Note that not every square matrix has an inverse, e.g. the all zero matrix.

HINT: If you want to multiply n by n matrices A and B , consider the $3n \times 3n$ matrix C of the following form:

$$\begin{bmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix}$$

Where I is the $n \times n$ identity matrix.

†

3. Show that if there is an $O(n^k)$, $k \geq 1$, time algorithm for squaring a degree n polynomial, then there is an $O(n^k)$ time algorithm for multiplying two degree n polynomials. Assume that the polynomials are given by their coefficients.

††

4. Consider the following variant of the minimum Steiner tree problem. The input is n points in the plane. Each point is given by its Cartesian coordinates. The problem is build a collection of roads between these points so that you can reach any city from any other city and the total length of the roads is minimized. The collection of roads should be output as an adjacency list structure, that is, for each point p of intersection of roads there are a list of roads that meet that point. Roads must terminate when they meet one of the original input points, or when they meet another road. For example, if the points are $(1, 1), (-1, 1), (1, -1), (-1, -1), (2, 2)$ the output would be:

From the point $(0, 0)$ there are roads to the points $(1, 1), (-1, 1), (1, -1)$, and $(-1, -1)$. From the point $(1, 1)$ there are roads to $(0, 0)$ and $(2, 2)$. From the point $(-1, 1)$ there is a road to $(0, 0)$. From the point $(1, -1)$ there is a road to $(0, 0)$. From the point $(-1, -1)$ there is a road to $(0, 0)$.

Roads from $(1, 1)$ to $(-1, -1)$, and from $(1, -1)$ to $(-1, 1)$ would not be allowed because they cross. A road from $(-1, -1)$ to $(2, 2)$ would not be allowed since it would cross the point $(1, 1)$.

Show by reduction that if you can solve this problem in linear time, then you can sort n numbers in linear time.

†

5. Show that if one of the following three problems has a polynomial time algorithm then they all do.
 - The Independent Set Problem: The input is a graph G . The problem is to find the largest independent set in G . In an independent set all vertices are mutually nonadjacent.
 - The Clique Problem: The input is a graph G . The problem is to find the largest clique in G . In a clique all vertices are mutually adjacent.

- The Vertex Cover Problem: The input is a graph G . The problem is to find the smallest vertex cover in G . A set S is a vertex cover if each edge in G is incident to a vertex in S .

Hint: These reductions all involve simple modifications to the graphs and the integer parameter.

††

6. Show that if one of the following three problems has a polynomial time algorithm then they all do.
- The problem is to determine whether a Boolean Circuit (with gates NOT, binary AND, and binary OR) has some input that causes all of the output lines to be 1. Assume that the fan-out (the number of gates that the output of a single gate can be fed into) of the gates in a circuit may be arbitrary.
 - The problem is to determine whether a Boolean Circuit (with gates NOT, binary AND, and binary OR), and fan-out at most 2, has some input that causes all of the output lines to be 1.
 - The problem is to determine whether a planar Boolean Circuit (with gates NOT, binary AND, and binary OR) has some input that causes all of the output lines to be 1. A circuit is planar if it can be laid on on the 2D plane so that no pair of lines cross (if you like, you can assume that the layout is part of the input).

HINT: There are two issues here: The first issue is to show that small fan-out does not make the circuit satisfiability problem any easier. To address this issue, you need to design fanout 2 circuitry that can simulate the large fanout of a particular gate. The second issue is to show that the satisfiability problem is no easier for planar circuits. To address this issue, you need to design to design planar circuitry to simulate the connection crossings in a nonplanar circuit.

†††

7. Show that if one of the following three problems has a polynomial time algorithm then they all do.
- The input is two undirected graphs G and H . The problem is to determine if the graphs are isomorphic.
 - The input is two directed graphs G and H . The problem is to determine if the graphs are isomorphic.
 - The input is two undirected graphs G and H , and an integer k . The problem is to determine if the graphs are isomorphic and all the vertices in each graph have degree k .

Intuitively, two graphs are isomorphic if one can name/label the vertices so that the graphs are identical. More formally, two undirected graphs G and H are isomorphic if there is a bijection f from the vertices of G to the vertices of H such that (v, w) is an edge in G if and only if $(f(v), f(w))$ is an edge in H . More formally, two directed graphs G and H are isomorphic if there is a bijection f from the vertices of G to the vertices of H such that (v, w) is a directed edge in G if and only if $(f(v), f(w))$ is a directed edge in H . The degree of a vertex is the number of edges incident to that vertex.

NOTE: There is no known polynomial time algorithm for the graph isomorphism problem. For reasons that are too complicated to go into here, it is unlikely that the graph isomorphism problem is NP-hard (but there is no known proof of this).

HINT: It is straight-forward to solve undirected graph isomorphism using a routine for directed graph isomorphism. The other direction is not so straight-forward. Your reductions here probably should introduce some additional pieces to the graphs so the only possible isomorphisms are of the type that you seek in the original problem.

†††

8. Show that the subset sum problem is self-reducible. The decision problem is to take a collection of positive integers x_1, \dots, x_n and an integer L and decide if there is a subset of the x_i 's that sum to L . The optimization problem asks you to return the actual subset if it exists. So you must show that

if the decision problem has a polynomial time algorithm then the optimization problem also has a polynomial time algorithm.

†

9. The input to the Hamiltonian Cycle Problem is an undirected graph G . The problem is to find a Hamiltonian cycle, if one exists. A Hamiltonian cycle is a simple cycle that spans G . Show that the Hamiltonian cycle problem is self reducible. That is, show that if there is a polynomial time algorithm that determines whether a graph has a Hamiltonian cycle, then there is a polynomial time algorithm to find Hamiltonian cycles.

†

10. Show that the clique problem is self-reducible. The decision problem is to take a graph G and an integer k and decide if G has a clique of size k or not. The optimization problem takes a graph G , and returns a largest clique in G . So you must show that if the decision problem has a polynomial time algorithm then the optimization problem also has a polynomial time algorithm. Recall that a clique is a collection of mutually adjacent vertices.

††

11. Show that the Vertex Cover Problem is self-reducible. The decision problem is to take a graph G and an integer k and decide if G has a vertex cover of size k or not. The optimization problem takes a graph G , and returns a smallest vertex cover in G . So you must show that if the decision problem has a polynomial time algorithm then the optimization problem also has a polynomial time algorithm. Recall that a vertex cover is a collection S of vertices with the property that every edge is incident to a vertex in S .

††

12. Consider the following 2Clique problem:

INPUT: A undirected graph G and an integer k .

OUTPUT: 1 if G has two vertex disjoint cliques of size k , and 0 otherwise.

Show that this problem is NP -hard. Use the fact that the clique problem is NP -complete. The input to the clique problem is an undirected graph H and an integer j . The output should be 1 if H contains a clique of size j and 0 otherwise. Note that a clique is a mutually adjacent collection of vertices. Two cliques are disjoint if they do not share any vertices in common.

†

13. Show that the following problem is NP -hard:

INPUT: A graph G . Let n be the number of vertices in G .

OUTPUT: 1 if G contains a simple cycle with \sqrt{n} edges, and 0 otherwise.

Use the fact the the following problem is NP -hard:

INPUT: A graph G .

OUTPUT: 1 if G contains a simple cycle that spans G , and 0 otherwise.

Note that a cycle is simple if it doesn't visit any vertex more than once. A cycle spans G if every vertex is included in the cycle.

†

14. Consider the problem where the input is a collection of linear inequalities. For example, the input might look like $3x - 2y \leq 3$ and $2x - 3x \geq 9$. The problem is to determine if there is an integer solution that simultaneously satisfies all the inequalities. Show that this problem is NP -hard using the fact that it is NP -hard to determine if a Boolean formula in conjunctive normal form is satisfiable.

†

15. Consider the following variant of the MST problem. The input consists of an undirected graph G and an integer k . The problem is to find a spanning tree T of G such that the degree of each node in T is at most k , or report that no such tree exists. Show by reduction that if this problem has a polynomial time algorithm then the Hamiltonian path problem has a polynomial time algorithm. The Hamiltonian path problem asks you to determine whether a graph has a simple path that spans the vertices.

†

16. The input to the three coloring problem is a graph G , and the problem is to decide whether the vertices of G can be colored with three colors such that no pair of adjacent vertices are colored the same color. The input to the four coloring problem is a graph G , and the problem is to decide whether the vertices of G can be colored with four colors such that no pair of adjacent vertices are colored the same color. Show by reduction that if the four coloring problem has a polynomial time algorithm then so does the three coloring problem.

†

17. Consider the following problem. The input is an undirected graph G and an integer k . The problem is to determine if G contains a clique of size k **AND** an independent set of size k . Recall that a clique is a collection of mutually adjacent vertices, and an independent set is a collection of mutually nonadjacent vertices. Show by reduction that if this problem has a polynomial time algorithm then the clique problem has a polynomial time algorithm

†

18. Consider the following problem. The input is an undirected graph G and an integer k . The problem is to determine if G contains a clique of size k **OR** an independent set of size k . Show by reduction that if this problem has a polynomial time algorithm then the clique problem has a polynomial time algorithm

††

19. Show by reduction that if the decision version of the SAT-CNF problem has a polynomial time algorithm then the decision version of the 3-coloring problem has a polynomial time algorithm.

†††

20. In the dominating set problem the input is an undirected graph G , the problem is to find the smallest dominating set in G . A dominating set is a collection S of vertices with the property that every vertex v in G is either in S , or there is an edge between a vertex in S and v . Show that the dominating set problem is NP -hard using a reduction from the vertex cover problem.

††

21. The input to the Fixed Hamiltonian path problem is an undirected graph G and two vertices x and y in G . The problem is to determine if there is a simple path between x and y in G that spans all the vertices in G . A path is simple if it doesn't include any vertex more than once. Show that if the Fixed Hamiltonian path problem has a polynomial time algorithm then the Hamiltonian cycle problem has a polynomial time algorithm.

HINT: I think it is easier to do this problem if you don't restrict yourself to a many-to-one reduction, that is, feel free to call the path procedure multiple times.

††

22. Consider the following problem. The input is a graph $G = (V, E)$, a subset R of vertices of G , and a positive integer k . The problem is to determine if there is a subset U of V such that

- All the vertices in R are contained in U , and
- the number of vertices in U is at most k , and
- for every pair of vertices x and y in R , one can walk from x to y in G only traversing vertices that are in U .

Show that this problem is NP-hard using a reduction from Vertex Cover. Recall that the input for the vertex cover problem is a graph H and an integer ℓ , and the problem is to determine whether H has a vertex cover of size ℓ or not. A vertex cover S is a collection of vertices with the property that every edge is incident on at least one vertex in S .

HINT: Let (H, ℓ) be the graph and integer pair instance for vertex cover. Have R consist of a single source vertex plus one vertex for each edge in H . Now have one more vertex in G , but not in R , for each vertex in H . Now add edges to G so that you can connect the vertices in R using few additional vertices if and only if H has a vertex cover of size ℓ .

† † †

23. In the disjoint paths problem the input is a directed graph G and pairs $(s_1, t_1), \dots, (s_k, t_k)$ of vertices. The problem is to determine if there exist a collection of vertex disjoint paths between the pairs of vertices (from each s_i to each t_i). Show that this problem is NP-hard by a reduction from the 3SAT problem. Note that this problem is not easy.

HINT: Construct one pair (s_i, t_i) for each variable x_i in your formula F . Intuitively there will be two possible paths between s_i and t_i depending on whether x_i is true or false. There will be a component/subgraph D_j of G for each clause C_j in F . There will be three possible paths between the (s_i, t_i) 's pairs for each D_j . You want that it is possible to route any two of these paths (but not all three) through D_j .

† † †

24. The input to the triangle problem is a subset W of the Cartesian product $X \times Y \times Z$ of sets X , Y and Z , each of cardinality n . The problem is to determine if there is a subset U of W such that 1) every element of X is in exactly one element of U , 2) every element of Y is in exactly one element of U , and 3) every element of Z is in exactly one element of U . Here's a story version of the same problem. You have disjoint collections of n pilots, n copilots, and n flight engineers. For each possible triple of pilot, copilot, and flight engineer, you know if these three people are compatible or not. Your goal is to determine if you can assign these $3n$ people to n flights so that every flight has one pilot, one copilot, and one flight engineer that are compatible. Show that this problem is NP-hard using a reduction from 3SAT.

† † †

25. We consider a generalization of the Fox, goose and bag of beans puzzle http://en.wikipedia.org/wiki/Fox,_goose_and_bag_of_beans_puzzle

The input is a graph G an integer k . The vertices of G are objects that the farmer has to transport over the river, there are an edge between two objects if they can not be left alone together on the same size of the river. The goal is to determine if a boat of size k is sufficient to safely transport the objects across the river. The size of the boat is the number of objects that the farmer can haul in the boat.

Show that this problem is NP-hard using a reduction from one of the problems that either I showed was NP-hard in class, or that you showed was NP-hard in the homework. So I am letting you pick the problem to reduce from here. You should take some time to reflect which problem would be easiest to reduce from.

††