

1. To reduce the problem of multiplying  $n \times n$  matrices  $A$  and  $B$  to the problem of multiplying two lower triangular matrices  $C$  and  $D$ , simply create a  $3n \times 3n$  matrix:

$$C = D = \begin{bmatrix} 0 & 0 & 0 \\ B & 0 & 0 \\ 0 & A & 0 \end{bmatrix}$$

then we have,

$$CD = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ AB & 0 & 0 \end{bmatrix}$$

The matrix  $AB$  can easily be obtained by isolating the lower left corner of the product matrix.

2. To reduce the problem of multiplying  $n \times n$  matrices to the problem of inverting a single matrix, construct a  $3n \times 3n$  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. By inverting  $C$  we get:

$$\begin{bmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix}^{-1} = \begin{bmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{bmatrix}$$

$AB$  can be easily obtained by extracting the upper right corner of  $C^{-1}$ .

3. Show that  $\text{PolyMult} \leq \text{PolySquare}$ .

Using the identity

$$(A + B)^2 = A^2 + B^2 + 2AB \text{ we have}$$

$$AB = [(A + B)^2 - A^2 - B^2]/2$$

Procedure PolyMult

```
{
Read A and B
x = PolySquare(A + B)    y = PolySquare(A)    z = PolySquare(B)
AB = (x - y - z)/2 }
```

The time for procedure Polymult is the time for procedure PolySquare plus  $O(n)$ .

4. To reduce the problem of sorting  $n$  numbers to the minimum Steiner tree problem, take the numbers  $x_1 \dots x_n$  to sort and create the points  $(x_1, 0) \dots (x_n, 0)$  in the plane. The minimum Steiner tree of these points is the tree that contains  $(x_1, 0) \dots (x_n, 0)$  in order.
5. To reduce of finding the subset that equals  $L$  to the problem of finding if such a subset exist, create a new set of  $x$ 's  $X'$  where  $X' = X - x_i$ . If there is not subset of  $X$  that equals  $L$ , return. Otherwise, if there is not subset of  $X'$  that is equal to  $L$ , then the answer-set = answer union  $x_i$  and  $L$ ,  $L = L$ -value( $x_i$ ). Recurse on  $X'$  for each  $x_i$  in  $X$ .
6. To reduce the problem of finding a Hamiltonian cycle in a graph  $G$  to the problem of determining whether one exists, create graph  $G'$  by removing an arbitrary edge  $(x, y)$ . Call the HAM-decision algorithm on both  $G$  and  $G'$ . If there is a Hamiltonian cycle in both, recurse on  $G'$ . If there is a Hamiltonian cycle in  $G$  but not  $G'$ , then we know  $(x, y)$  is in the Hamiltonian cycle. Recurse on  $G$ , taking care not to select edge  $(x, y)$  again.
7. To reduce the clique optimaization problem to the clique decision problem, we need to show that problem of finding a clique of maximum size is reducible to determine the size of the maximum clique. Algorithm for finding clique: Pick a vertex  $v$ . If the maximum clique size in  $G - v$  is equal to the maximum clique size in  $G$ , we can recurse on  $G - v$ . Otherwise, add  $v$  to final clique, delete  $v$  and vertices in  $G$  not adjacent to  $v$ , and recurse.
8. No solution provided.
9. To reduce the clique problem to the 2clique problem, create a new graph  $G'$  by adding a  $k$ -vertex clique to  $G$ ; do not connect any of the vertices in the new clique to any of the vetices of the original graph. Call the procedure for 2clique on  $G'$ ; a 1 will be returned if and only if  $G$  has a clique of size  $k$ .
10. We reduce the Hamiltonian path problem to our new problem. Let  $G$  be the input graph to the Hamiltonian path problem. Let  $n$  denote the number of vertices in  $G$ . We then create a new graph  $H$  that consists of  $n$  disjoint copies of  $G$ . Note that  $H$  has  $n^2$  vertices. Then if  $H$  has a simple cycle with  $\sqrt{n^2} = n$  vertices then this simple cycle must lie entirely within on of the copies of  $G$ . Thus  $G$  has a simple cycle spanning all of its vertices if and only if  $H$  has a simple cycle spanning the square root of the number of its vertices.
11. To reduce the Hamiltonian path problem to the  $k$ -degree MST problem, check to see if there is a MST with degree at most 2 (such a tree would be a Hamiltonian path).

12. To reduce the 3-coloring problem to the 4-coloring problem, create a new vertex  $v$  that is connected to every other vertex in the original graph. Since  $v$  is connected to every vertex, it must have a different color than the others. This new graph can be colored with 4 colors if and only if the original graph can be colored with 3 colors.
13. To reduce the clique problem to the clique/independent set problem, add to  $G$  an independent set of size  $k$  to get a graph  $G'$ . Then  $G'$  has a clique of size  $k$  and an independent set of size  $k$  if and only if  $G$  has a clique of size  $k$ .
14. To reduce the 3-coloring decision problem to the SAT-CNF: Assume the colors are red, blue, and green. For each vertex  $x$  in  $G$ , create the following variables:  $x_{red}$ ,  $x_{blue}$ , and  $x_{green}$ . For each vertex  $v \in G$ , exactly one of  $v_{red}$ ,  $v_{blue}$ , and  $v_{green}$  must be true, and for each vertex  $u$  adjacent to  $v$ , the  $u$ -variable corresponding to  $v$ 's color must be false (that is,  $\bar{v}_{color} \vee \bar{u}_{color}$  must be true). The construction of the entire clause is straightforward; simply "and" together all the vertex clauses. If there is an assignment to these variables that makes the entire clause true, the graph can be colored with 3 colors.
15. To reduce the vertex-cover problem to the dominating set problem, create a new graph  $G'$  by replacing each edge  $e = (x, y)$  in  $G$  by a new vertex  $v_e$  and edges  $(x, v_e)$ ,  $(y, v_e)$  and  $(y, x)$ . The vertex  $v_e$  essentially forces the edge  $e$  to be covered. Call the procedure for dominating set to get a minimum cardinality dominating set  $D$ . If a  $v_e \in D$  then replace  $v_e$  by  $x$  (or  $y$  doesn't matter). Call this new set  $D'$ . Note that  $D'$  is still dominating in  $G'$ . Then  $D$  is a minimum cardinality vertex cover in  $G$ . (Note this reduction only handles the case when  $G$  is a connected graph and has at least one edge. If  $G$  is not connected, run this reduction on each non-trivial connected component.)
16. To reduce the Hamiltonian Cycle problem to the Fixed Hamiltonian path problem, check if there is a Hamiltonian path between  $x$  and  $y$  for each edge  $e = (x, y) \in G$ .
17. Note that  $G$  has an independent set of size  $k$  if and only if the complement of  $G$  has a clique of size  $k$ . The complement of a graph  $G$  contains an edge between vertices  $x$  and  $y$  if and only if  $G$  does not contain an edge between  $x$  and  $y$ . Thus independent set and clique are reducible to each other by complementing the graph and calling the procedure for the other problem.

Note that  $G$  has an independent set of size  $k$  if and only if  $G$  has a vertex cover of size  $n - k$ . Here  $n$  is the number of vertices in  $G$ . One can prove this by noting that the vertices not in a vertex cover form an independent

set, and the vertices not in an independent set form a vertex cover. Thus the problems of computing the size of the minimum vertex cover and the size of the maximum independent set are obviously polynomial time equivalent. Since both problems are decision reducible, the optimization problems are also polynomial time equivalent.

18. [This solution was adapted from Brian Wongchaowart's homework writeup.]

To solve the vertex cover problem for an undirected graph  $G$  and an integer  $m$ , construct a new graph  $H$  to be used as input to the minimum tree problem. Begin by adding a new vertex  $r$  to  $H$ . Then for each vertex  $v_k$  in  $G$ , add a vertex to  $H$  labeled  $v_k$  and an edge between this vertex and  $r$ . For each edge  $v_i, v_j$  in  $G$ , add a vertex to  $H$  labeled  $e_{ij}$  and add an edge between this vertex and the vertices in  $H$  labeled  $v_i$  and  $v_j$ .

Add vertex  $r$  and all of the vertices  $e_{ij}$  corresponding to edges of  $G$  to the set  $R$ . Let  $e$  be the number of edges in  $G$ . Use  $H$ ,  $R$ , and  $m + e$  as the input to the minimum tree problem. A tree containing all of the vertices in  $R$  and at most  $m + e$  edges is a subgraph of  $H$  if and only if  $G$  has a vertex cover of size  $m$ :

( $\Leftarrow$ ) If  $G$  has a vertex cover  $C$  of size  $m$ , such a tree can be found in  $H$  by taking  $r$  as the root, adding the  $m$  vertices labeled with the vertices in  $C$  as  $r$ 's children, and finally adding each of the  $e$  vertices corresponding to an edge in  $G$  as a leaf of the tree. Each leaf is attached to the tree by using the edge connecting it to one of  $r$ 's children (such an edge must exist by the definition of a vertex cover). This tree has  $m + e$  edges.

( $\Rightarrow$ ) If, on the other hand,  $H$  contains a tree  $T$  that includes all of the vertices in  $R$  and has at most  $m + e$  edges, the vertices in  $G$  that correspond to a vertex in  $T$  form a vertex cover for  $G$  of size at most  $m$ . To see why, let the root of  $T$  be  $r$  (which must be in  $T$  by definition of  $R$ ). Note that  $r$  is not connected directly to any vertex  $e_{ij}$  in  $H$ , but by definition of  $R$ , these  $e$  vertices must all be in  $T$ . So  $T$  must include as  $r$ 's children some number of the vertices in  $H$  such that there is an edge from one of these vertices to each vertex  $e_{ij}$ . This corresponds exactly to choosing a set of vertices in  $G$  so that each edge is incident to a vertex in the set, i.e., choosing a vertex cover. The number of  $r$ 's children, the size of this vertex cover, cannot be greater than  $m$  because  $e$  edges out of the at most  $m + e$  edges in  $T$  are used to connect  $r$ 's children to the  $e_{ij}$  vertices.

This reduction can clearly be carried out in polynomial time, as one vertex is added to  $H$  for each vertex and each edge in  $G$ .

19. [This solution was adapted from Brian Wongchaowart's homework writeup.]

Suppose that  $F$  is a Boolean formula in CNF with  $k$  clauses and exactly three literals per clause. It is possible to construct a directed graph  $G$  with pairs of vertices  $(s_1, t_1), \dots, (s_k, t_k)$  such that the disjoint paths problem



$v_j$  in the gadget. In the disjoint paths problem, if  $C_j$ 's path passes through  $v_j$ , it becomes impossible for  $C_i$ 's path to pass through this gadget. In this way a solution to the disjoint paths problem for  $G$  and  $(s_1, t_1), \dots, (s_k, t_k)$  cannot make use of two opposing literals  $x$  and  $\bar{x}$  in different clauses (i.e. cannot assign  $x$  to be both true and false).

Thus it is easy to see that a solution to the disjoint paths problem can be turned into a satisfying assignment for  $F$  and that a satisfying assignment for  $F$  allows one to pick disjoint paths in  $G$ . This reduction can clearly be carried out in polynomial time, as  $G$  contains two vertices for each clause in  $F$  and a gadget for each literal, with the size of the literal gadgets being proportional to the number of clauses.