In order to compare the hardness of solving different problems, we use reductions. The idea is that if I can show that I can solve a problem by using a black box that solves B, then I can understand the difculty of solving a problem in terms of the work required to transform a solution of B to the solution of a problem. In this case, we consider a polynomial time algorithm to be efcient. We consider the set of all decision problems for which there exists a Polynomial-time verication algorithm.

A decision problem is NP-Hard if all problems in NP are polynomial time re-ducible to it. That is, given an efcient algorithm which solves an NP-hard problem, we can easily construct an algorithm for a decision problem. In other words, if solving B is easy then solving A is easy. Equivalently, if A is hard, then B is hard.

Given a directed graph, is there a cycle that visits every vertex exactly once? Given that Hamiltonian Cycle is NP-Complete, we prove that Hamiltonians Path is NP. To prove this, we need to prove that there exists a verier Vx, y.

Suppose that we are given a path that traverses every vertex of the graph. We can use this to check that the path traverses all the vertices in the graph exactly once and then check that every edge in the path is an edge in a graph. In this case, we can use the path to ensure that all the edges in that path are an edge of thegraph.

Hamiltonian Cycle is a combinatorial problem that is hard to solve. We prove this by giving a Karp-reduction of Hamiltonian Cycle to Hamiltonian Path. We use this to show that Hamiltonian path is NP-Hard. We define a graph that consists of two vertices and each vertex is traversed exactly once.

Suppose that there is a Hamiltonian cycle in a given Hamiltonian path. We can use the edges of the cycle as the path on the path but the path must begin on v and end on v. The transformation takes

at most OE. If there are two Hamiltonian cycles in the same cycle, there is one Hamiltonian Cycle on the other Hamiltonian Path. The two cycles must begin at the beginning and end at the end of the path. The path must start at the start and end point.

Suppose that there are three disjoint sets X, Y, Z, which each contain n elements and triples. Given that 3Dimensional Matching is NP-complete, we prove that there is a partition into four subsets of four elements each with the same sum. We can verify a potential solution by checking that the elements in each partition sum to t, and that no element is used more than once.

Suppose that given an input X, Y, Z and triples T to 3DM, we create a set of n integers as part of our set of integers. This takes at most On2 naively. We prove this by giving a Karp-reduction of 3DM to 4-partition.

Suppose that we want to set a target sum of 40r415 for each triple in the triple equation. We set our target sum by adding a triple element with three actual elements or three dummy elements. The target sum can be achieved by adding three actual triple elements or by using a dummy element. We can then set the target sum to 40R415.

Suppose that we are given a set of four elements that contains a triple element and three dummy elements. We are given the sum of the three actual elements and the three dummy ones that will sum to 40r4 15. We can then construct a four-partition solution. For every triple xi, yj and zk, we can create a 4 element set.

Suppose we are given a solution to the problem of four-partition. We would like to determine the sum of the element sizes of any four element set in the solution. By considering the sum, we can show that the set contains one element corresponding to each of the members in a triple and that all three elements are either actual elements or dummy elements. If the triple contains only actual

elements, it is part of the 3DM solution, otherwise it is not. This is possible only if the four elements correspond to a triple element and one element each from X, Y, and Z.

Suppose we have a graph that contains a set of vertices that is complete. We would like to reduce the graph to an independent set. Given that the graph is NP-complete, we need to prove that the independent set is a complete set. We can then reduce the partition to a partition that is at least as hard as the partition of the graph.

Suppose that given an input x G, k to Clique, create a set of vertices that has the same vertices but has edge u, v if and only if u,v E. This takes OE time, so it takes polynomial time to check that x is a valid input. Therefore, Independent Set is in NP. We prove this by giving a Karp-reduction of Clique to Independent Set.

Suppose that given a set of n elements in the graph G and an integer k, does there exist any set of k sets that are adjacent to each other? This shows that there are k elements that are not adjacent to one another in the construction of the graph. We can then reduce the construction to a set that is NP-complete.

Suppose that we are given the set of all edges in the union. We can determine whether or not all all elements of the union are in the same union. To prove this, we reduce Vertex Cover to Set Cover. To do this we reduce the Vertex cover to set Cover. In Ok time, we can determine if or not we have exactly k sets.

Suppose that there is a k-Vertex Cover. If the vertex is part of the vertex cover, then the set cover. If there is no k-Set Cover, then there are two sets of vertices. This means that every edge is incident to some vertex. Since there are vertices, every element ej S is covered by the set. This new input is polynomial because each set has size at most E and there are V sets. This is because every edge

ej E is incident with some vertex, which means that the set Si is covered.

Suppose that given a CNF formula consisting of clauses C1, c2, cn and lit erals X x1, x2, xk such that each clause involves exactly two literals, does there exist an assignment of TRUEFALSE to the literals such that at least k clauses of the formula are satised? To prove this, we need to prove there exists a polytime verier. The witnesscerticate is an assignment to the Literals that satises at least one of the clauses. A polytime Verier can simple evaluate each clause of the cNF formula given the assignment, and

Suppose that given an input G, k to Clique, we create an input such that the input C, X, k is given such that yes instances of Clique map to no instances of Max2SAT. To prove this, we reduce Clique to Max2sAT. We then create an output such that all instances of the input G and k are given the same input as the input X and k. We can then use this output to create the output C, x and k to produce the output of the output X.

In designing the CNF formula, we need to design clauses that act as constraints to enforce that x1 True corresponds to vertex 1 being chosen in the clique. Recall that a set of vertices U is a clique if for all i, j U, i, J E. Equivalently, U is cliqued if for every non-edge such that i, V such that J E, either i U or J U. We will use these constraints to design the clauses in our CNF formulas.

In order to encourage choosing cliques with more vertices, for every vertex i we introduce the clauses xi z and z, where z is a new literal. This means that for every non-edge, at least one of the endpoints must not be in the clique. However, these clauses could also be satised by setting all literals to FALSE.

Suppose that the input to Max2SAT is dened by a series of literals. In order to satisfy these two clauses, xi must be true. Choose k number of non-edges. If xi is true, then the input must be equal

to the number of vertices.

Suppose that given a clique in a graph such that we can use an assigment such that for all the cliques in that graph, we can then use the assigments such that if the clique is a certain subset of the graph, then Max2SAT is true. Show that if we are given a certain clique, we will be able to use the Assigment to determine the number of cliques that exist in the graph.

In this lecture, we discuss the assigment that satises at least k clauses in the CNF formula. This is a statement that states that for all non-edges in the formula, the clause xi xj is satised because U is a clique. For all i U, both xi z and xj are satised.

Suppose that given an assignment for the literals X such that at least k clauses are satised, we will show that if Max2SAT is yes, then the number of satised clauses is non-edges. If for all i, j E, xi xj is satised and xi z is not satised. Therefore, we can show that we can nd a k-clique in the original graph that has size k by construction.

If the assignment does not correspond to a clique, then we will show that we can modify the assignment to nd aClique of size at least k. We continue to choose unsatised non-edge clauses and modifying the assignment in this way until all the non-edges clauses are satised. Note that the number of unsatisation clauses is monotonically decreasing in each modication. Therefore, we obtain an assignment which corresponds to the clique. In addition, we can also obtain a assignment that corresponds to an assignment that contains a size of k. Since every modication does not increase or stay the