Reductions between np complete biology essay

Science, Biology

problemsAdam JamaAbstractThis is where an abstract can appear. If you don't have an abstract, just throw out theses lines. The abstract should not exceed 10 lines :-)345678910Project Dissertation submitted to the University of Wales, Swanseain Partial Ful_lment for the Degree of Bachelor of Science1Department of Computer ScienceUniversity of Wales SwanseaDeclarationThis work has not previously been accepted in substance for any degree and isnot being currently submitted for any degree. May 2, 2013Signed: Statement 1This dissertation is being submitted in partial ful_lment of the requirements forthe degree of a BSc in Computer Science. May 2, 2013Signed: Statement 2This dissertation is the result of my own independent work/investigation, exceptwhere otherwise stated. Other sources are speci_cally acknowledged by clearcross referencing to author, work, and pages using the bibliography/references. I understand that failure to do this amounts to plagiarism and will be consideredgrounds for failure of this dissertation and the degree examination as a whole. May 2, 2013Signed: Statement 3I hereby give consent for my dissertation to be available for photocopying andfor inter-library loan, and for the title and summary to be made available tooutside organisations. May 2, 2013Signed: Contents

Chapter 1Introduction1. 1 IntroductionIn Computational Complexity the class NP-complete is a very important researcharea in Computer Science, reason being that determining whether or notevery problem whose solution can be e_ciently veri_ed by a computer, can alsobe quickly solved by a computer. The following problem is referred to asthe P versus NP problem, which is one of the principal unsolved problemsin Computer Science to date. Also, demonstrating that many computationalproblems that occur throughout Computer Science are in the complexity classNP-complete, by showing a transformation from a known NP-complete problem. The main purpose of this dissertation is to study the Computational Complexityclass NP-complete and in particular the reduction notion. This paperwill focus on 8 decision problems which are known to be in the complexity classNP-complete and show their reductions from known NP-complete problems. For the 8 NP-complete problems I will give the precise de_nitions. I will alsogive examples when the decision problem is satis_ed and in the case its not. I will start by introducing the Computational Complexity classes P, NP andNP-complete. For each of the complexity classes, I will begin by giving a briefinsight into the history behind the complexity classes. Also, providing precisede_nitions of the complexity class, referring to known problems that are in theclass and inserting illustrations when needed. Subsequently, I will present the precise de_nition of the 8 problems also takinginto account the di_erent interpretations of the problem. I will go onto givea brief insight into the history behind the problems. Also, providing examples ofapplications that model these 8 problems and insert illustrations when needed. Finally, I will de_ne each of the 8 decision problem by giving the input, sizeof the problem and output. We will give the history of discovery of each problem, show where the NP-completeness of the decision problem was _rst shown. Firstly we verify that each of the decision problems can be e_ciently veri_ed byproviding examples. We then derive rigorous proofs for each reduction, provingthat the decision problem is in the complexity class NP-complete. 61. 2 DiscoveryIn 1971, Stephen Cook a Canadian Computer Scientist introduced the theoryof NP-completeness, in a paper entitled " The Complexity of Theorem ProvingProcedures" [7]. Within this paper Stephen Cook introduced importantconcepts regarding NP-completeness. Firstly, he con_rmed the class NP, were the abbreviation of NP refers tonondeterministic polynomial time. The decision problems in the class NP canbe solved in polynomial time by a nondeterministic computer. Also, he stressed the importance of polynomial time reduction, such thatevery problem in the decision class NP is reducible to a given decision problemC in polynomial time. If it's the case, that we have a polynomial time reductionfrom one NP problem to another given NP problem. This con_rms thatany polynomial time algorithm for the second problem can be converted into apolynomial time algorithm for the _rst problem. Thirdly, he introduced the _rst decision problem in the class NP, the Satis-_ability problem which is more often than not referred to as the SAT problem. This con_rms that a given decision problem c in NP, can be polynomially reducedto the Satis_ability problem. If the satis_ability problem can be solvedwith a polynomial time algorithm, then so can every problem in NP, and if anyproblem is in NP-hard, then the satis_ability problem also must be hard. Finally, Stephen Cook proved that a problem in the decision class NP mayalso be in the the decision class NP-hard such that, a given decision problem c isNP-hard if and only if there is a known problem in NP-complete i. e the problemthat is polynomial time reducible to c. Subsequently, in 1972 Richard Karp an American Computer Scientist developedon the ground work of Stephen Cook by using the Boolean Satis_abilityproblem to prove several other problems were in the class NP-complete by showingthere is a polynomial-time reduction from the Boolean Satis_ability problemto each of the Karp's 21 NP-complete problems in his paper, " Reducibility AmongCombinatorial Problem" [19]. The development of Richard Karp's 21NP-complete problems steered a wide-ranging interest in the concept of NPcompleteness. 7Chapter 2Preliminaries2. 1 Graph theoryDe_nition 2. 1. 1 A graph is a pair (V; E), where V is the set of vertices, andE is the set of edges, where an edge is a 2-element subset of V . Often a graphis denoted by G = (V; E), and we may use V (G) := V and E(G) := E. [14]Example 2. 1. 2 Some simple examples for graphs, using mathematical nota-tion, and additionally drawing them XXX1. A graph with one vertex: (f1g; ;). Note that a graph with at most onevertex can not have an edge. 2. There are exactly two possible graphs G with V (G) = f1; 2g, namely withE(G) = ; or E(G) = ff1; 2gg. 3. There are exactly three possible graphs G with V(G) = f1, 2, 3g, namelywith E(G) = ; or E(G) = ff1, 2gg or E(G) = ff1, 2g, f2, 3g, f3, 1ggFigure 2. 1: Undirected graphs. (a) A undirected graph G = (V, E), where V = f; g and E = f; g. (b) An undirected graph G = (V, E), where V = f1g and E= f; g. (c) An undirected graph G = (V, E), where V = f1, 2g and E = f(1, 2)g.

De_nition 2. 1. 3 A bipartite graph is an undirected graph G = (V; E), inwhich V can be partitioned into two sets V1 and V2 such that (u, v) 2 E implieseither u 2 V1 and v 2 V2 or u 2 V2 and v 2 V1. That is, all edges go betweenthe two sets V1 and V2 [25]. 8Example 2. 1. 4 Some simple examples for bipartite graphs, using mathematicalnotation, and additionally drawing them XXX1. The bipartite graph G = (5, 6), where V1 = f1, 2, 3g, V2 = f4, 5g and E = f(1, 4),(1, 5),(2, 4),(2, 5),(3, 4),(3, 5)g2. The bipartite graph G = (9, 8), where V1 = f1, 2, 3, 4, 5g, V2 = f6, 7, 8, 9gand E = f(1, 6),(2, 6),(2, 7),(3, 8),(3, 9),(4, 7),(5, 6),(5, 9)gFigure 2. 2: Undirected graphs. (a) A undirected graph G = (V, E), where V1= f1, 2, 3g and V2 = f4, 5g and E = f(1, 4),(1, 5),(2, 4),(2, 5),(3, 4),(3, 5)g. (b) Anunidirected graph G = (V, E), where V1 = f1, 2, 3, 4, 5g, V2 = f6, 7, 8, 9g and E = f(1, 6),(2, 6),(2, 7),(3, 8),(3, 9),(4, 7),(5, 6),(5, 9)g. 2. 2 SetsDe_nition 2. 2. 1 A Set, is a collection of distinguishable objects, called itselements. If object x is a element of set S, we write x 2 S. If x 62 S. We expressa set by listing all the elements inside braces. Example 2. 2. 2 Some simple examples for sets, using mathematical notations, and additionally drawing them: 1. The set S = f20, 95, 106, 48, 52, 7g. The set S contains the integers 7, 20, 48, 52, 95and 106. 2. The set S = f32, 77, 21, 54, 21, 77g. The set S contains the integers 32, 77, 21, 54, 21and 77. 3. The set S = f1, 2, 3, 4, 5, 6, 7, 8, 9g. The set S contains the integers 1, 2, 3, 4, 5, 6, 7, 8and 9. 9Figure 2. 3: Finite sets. (a) The _nite set S = f20, 7, 48, 52, 106, 95g. (b) The _niteset S = f37, 106, 52, 95, 48, 7g. (c) The _nite set S = f1, 9, 8, 3, 6, 7, 2, 5, 4g. Example 2. 2. 3 Some special notations for frequently encountered sets, usingmathematical notations: 1. ; expresses the empty set, which is the set containing no elements. 2. Z expresses the integer set, which is, the set f...,-2,-1, 0, 1, 2,... g3. R expresses the real number set, which is, the set4. N expresses the natural number set, which is, the set f0, 1, 2,... g. De_nition 2. 2. 4 A subset, is a collection of distinguishable elements, suchthat, all the elements of the set A are in the set B, if x 2 A implies x 2 B. Weexpress a A is a subset of B by writing A _ B. Example 2. 2. 5 Some simple examples for subsets, using mathematical nota-tions, and additionally drawing them: 1. The _nite sets S = f1, 3, 40, 20, 3, 1g and S' = f40, 20, 3g show that S' _ S2. The _nite sets S = f7, 2, 7, 9, 1g and S' = f9, 1g show that S' _ S3. The _nite sets S = f2, 3, 8, 14, 11g and S' = f14, 11g show that S' _ S10Figure 2. 4: Finite sets. (a) The _nite set S' = f40, 20, 3g is a subset of theset S = f1, 3, 40, 20, 3, 1g. (b) The _nite set S' = f9, 1g is a subset of the setS = f1, 7, 9, 77, 2, 7g. (c) The _nite set S' = f11, 14g is a subset of the set S = f2, 14, 8, 3, 11g2. 3 LogicDe_nition 2. 3. 1 In Boolean Logic, a formula is in Conjunctive Normal Form(CNF) if it is of the form CiV

...

VCn where each Ci is a disjunction of theliterals. The formula Ci are also called (disjunctive) clauses. Example 2. 3. 2 Some simple examples for CNF Boolean Logic, using mathe-matical notation, and additionally drawing them: 1. The following formula is a disjunction of two literals AWB. 2. The following formula is a conjunction of two clauses AVB. 3. The following formula is a conjunction of two clauses where each clausehas a disjunction of literals (AWB)V(CWB). De_nition 2. 3. 3 A Boolean Logic formula is said to be in n-CNF, where n isa natural number, if it is in CNF and every clause contains at most n literals. 11Figure 2. 5: Boolean Logic formulas. (a) The truth table for the ConjunctiveNormal form clause (AWB). (b) The truth table for the conjunction of twoclauses AVB. (c) The truth table for the Conjunctive Normal formula, whereeach clause has two distinct literals (AWB)V(CVB). This shows that the third CNF example above is in 2-CNFDe_nition 2. 3. 4 In Boolean Logic, a formula is in Disjunctive Normal Form(DNF) if it is of the form CiW

...

WCn where each Ci is conjunction of literals. The formula Ci are also called (conjunction) clauses. Example 2. 3. 5 Some simple examples for DNF Boolean Logic, using mathe-matical notations, and additionally drawing them: 1. The following formula is a conjunction of two literals: AVB2. The following formula is a disjunction of two clauses: AWB3. The following formula is a disjunction of two clauses where each clausehas a conjunction of literals: (AVB)W(CVB)12Figure 2. 6: Boolean Logic formulas. (a) The truth table for the DisjunctiveNormal form clause (AVB). (b) The truth table for the disjuntion of twoclauses AVB. (c) The truth table for the Disjunctive Normal formula, whereeach clause has two distinct literals (AVB)W(CVB). De_nition 2. 3. 6 The DeMorgan's Law1. :(PWQ) ! (: P)W(: Q)2. :(PVQ) ! (: P)V(: Q)In other words, the DeMorgan's Laws are transformation rules, such that thenegation of a conjunction is the disjunction of the negations. Als, the negationof a disjunction is the conjunction of the negations. 2. 4 Polynomial-timeDe_nition 2. 4. 1 In Computational Time Complexity, a given algorithm is aPolynomial Time algorithm if on inputs of size n, their worst-case running timeis O(nc). Example 2. 4. 2 Some simple examples of Polynomial Time algorithms, usingmathematical notaion: 1. DIV(x, y) = (q, r) such that x = y. r + r wherere r < y. The remainder ris also denoted x mod y. 2. GCD is de_ned as the largest z such that z/x and z/y. 3. EXP(x, y) = xy. 132. 5 FunctionsDe_nition 2. 5. 1 For any real number x, we denote the greatest integer lessthan or equal to x by bxc " the oor of x". De_nition 2. 5. 2 For any real number x, we denote the least integer greaterthan or equal to x by dxe " the ceiling of x". Example 2. 5. 3 Some simple examples using the oor and ceiling function: 1. x = 2. 4: bxc = 2 and dxe = 32. x = 2. 9: bxc = 2 and dxe = 33. x = 3: bx/2c = 1 and dx/2e = 24. x = 7: bx/2c = 3 and dx/2e = 42. 6 MatricesDe_nition 2. 6. 1 In mathematics, a matrix is a rectangular array of number, symbols and expressions, which are arranged in rows and columns. Example 2. 6. 2 Some simple examples using matrices: 1. x = 2. 4: bxc = 2 and dxe = 32. x = 2. 9: bxc = 2 and dxe = 33. x = 3: bx/2c = 1 and dx/2e = 24. x = 7: bx/2c = 3 and dx/2e = 414Chapter 3P, NP and NP-Complete3. 1 OverviewIn chapter 3, we present the computational complexity classes P, NP and NPcomplete. I will begin the chapter by introducing the complexity class P, whichis the set of decision problem which can be solved e_ciently. I begin this sectionby giving an insight into the history behind the notion P. I will then go ontopresent the formal de_nition of the Complexity class P and provide examplesof computational problems that are in the complexity class P. Subsequently, I will go onto introduce the complexity class NP, which is theset of decision problems whose solution can be veri_ed e_ciently by a computer. I also begin this section by giving an insight into the history behind the notionNP. We go onto to present the formal de_nition of the complexity class NPby showing the two conditions that must hold if the decision problem has ae_ciently veri_able proof system. I will also provide some examples of decisionproblems that are in the complexity class NP.

Finally, I will conclude this chapter by introducing the complexity classNP-complete, which is the set of decision problems whose solution cannot besolved e_ciently by a computer. In this section we present the theory of NPcompleteness, which is based on the notion of a polynomial time transformation. We present the formal de_nition of the theory of NP-completeness, by showingthe steps an NP problem has to take in order to to be in the complexity classNP-complete. I will then go onto provide the formal de_nition of a polynomialtime reduction and give examples of well known reductions of NP-completeproblems. 3. 2 P3. 2. 1 DiscoveryIn 1964, Alan Cobham an American mathematician introduced the set of problemswhich are e_ciently solvable in polynomial time, in his paper entitled " Theintrinsic computational di_culty of functions" [5]. He proved that many of themathematical functions can be proved in polynomial time. 15Subsequently, in 1965, Jack Edmond a Canadian mathematician proved themaximum matching problem has a polynomial time algorithm named the Blossomalgorithm, in his paper entitled " Paths, trees, and owers" [9]. De_nition 3. 2. 1 A decision problem S _ f0, 1g_ is e_ciently solvable if thereexists a polynomial-time algorithm A such that, for every x, it holds that A(x)= 1 if and only if x 2 S [24]. In other words, a decision problem S is in the complexity class P if thereexists a polynomial time algorithm which solves S. Example 3. 2. 2 Some examples of decision problems that are in complexityclass P: 1. Primality - Given an integer n > 0, determining whether or not n is primewas a commonly probed question in mathematics. Since, Carl Gauss, aGerman mathematician challenged mathematicians to solve primality ef-_ciently. The achievement of _nding an e_cient algorithm for primalitywas well presented by Agrwal and Mainindra in the paper entitled " Primesis in P" [1]. 2. GCD - Given two integers a and b (where at least one of which is non-zero), dertmine the largest integer z which divides a and b. The mostfamous algorithm for solving the Greatest Common Divisor is the EuclidAlgorithm which is shown in [20]. 3. Maximum matching - Given a graph G = (V, E), a Maximum Match m inG is a set of pairwise non-adjacent edges, such that no two edges share acommon vertex. Jack R. Edmonds a Canadian mathematician e_cientlysolved the maximum match problem, in a paper entitled " Paths, tree andowers" [9]. 4. Eulerian Cycle - Given a graph G =(V, E), a Eulerian Cycle c in G isa closed cycle that spans through all the edges of G. The Eulerian cycleof given graph can be solved e_ciently by using the Fleury's Algorithm. Wilson and Robin [28] describe the Fleury's algorithm well. 5. Connectivity - Given a grah G = (V, E), a given graph G is connectedif there is a path between every pair of vertices. The connectedness of agiven graph G can be solved e_ciently by either using depth-_rst searchor breadth-_rst search algorithms. Targan and Hopcroft [16] describe thealgorithm well for determining whether or not a graph is connected. 16Figure 3. 1: Given an undirected graph G. (a) A graph G = (V, E), where V = f1, 2, 3, 4g and E = f(1, 2),(2, 3),(3, 4),(4, 2)g. (b) A graph G' = (V, E), where V= f1, 2, 3, 4g and E = f(1, 2),(2, 3),(3, 4),(4, 2)g, such that f(1, 2)g and f(3, 4)g donot share a common vertex. 3. 3 NP3. 3. 1 DiscoveryIn 1971, Stephen Cook a Canadian Computer Scientist introduced the theoryof NP-completeness in his paper " The Complexity of Theorem Proving" [7], which provided the _rst NP-completeness proof. Stephen cook proved the _rstproblem in the class NP, the Satis_ability problem, which is more often thatnot referred to as the SAT problem and also proved the same for 3SAT. Subsequently, in 1972 Richard Karp an American Computer Scientist presentedhis paper " Reducibility among Combinatorial Problems" [19]. StephenKarp's paper introduced 19 more problems that were NP-complete, which includethe Hamiltonian Circuit, Clique and Vertex cover. It also provided methodsto prove NP-completeness using transformations from problems known tobe NP-complete. In 1979, Michael Garey and David Johnson published the textbook entitled" Computers and Intractability: A Guide to the Theory of NP-completeness"[14]. This was the _rst book on the theory of NP-completeness. Garey andJohnson provide countless number of problems that are NP-complete and providethe original sources where its NP-completeness was shown. De_nition 3. 3. 1 A decision problem S _ f0, 1g* has an e_eciently veri_ableproof system if there exists a polynomial p and a polynomial-time (veri_cation)algorithm V such that the following two conditions hold: 1. Completeness: for every x 2 S, there exists y of length at most p(jxj) suchthat V(x, y) = 1. 2. Soundness: For every x = 2 S and every y, it holds that V(x, y) = 0. Thus, x 2 S if and only if there exixts y of length at most p(jxj) such that V(x, y)= 1. In such a case, we say that S has an NP-proof system, and refer to V as itsveri_cation procedure (or as the proof system itself). We denote NP theclass of decision problems that have e_ciently veri_able proof systems [24]. 17In other words, condition one refers to true assertions that have valid proofs. Assertions refer to elements in S, from this we understand for every x belongingto S exists a string y such that V(x, y) = 1 (YES). v accepts y as a valid prooffor the elements of x in S. Conversely, condition two refers to false assertions that have not valid proofs, that is for every x not belonging to the set S and every string y it holds thatV(x, y) = 0 (NO) v rejects y as a proof for the elements of x in S. Example 3. 3. 2 Some examples of decision problems that fall into the class NP: 1. Set Cover - Consider a collection C of subsets of a _nite set S and a pos-itive integer K _ jCj. Does C contain a cover of S of size K. Consider a _-nite set S = f5, 6, 7, 8, 9g and set of sets C = ff5, 6, 7, 8, 9g, f6, 8g, f7, 8g, f8, 9ggand K = 2. V((S, C), 2) where S is the _nite set and C is the set of setsn, whose union is S, Thus for K = 2 we have ff5, 6, 7gf8, 9gg is a coverfor S. 2. Directed Hamiltonian Circuit - Consider a directed graph G (recall De_-nition 5. 4. 1). Does G contain a directed Hamiltonian circuit. A directedgraph G = (V, E) = f1, 2, 3, 4g and E = f(1, 2),(2, 5),(2, 4),(2, 2),(5, 4),(4, 5),(4, 1)g. V(G, f1, 2, 3, 4, 1g) we _nd we can derive a Directed Hamiltonian cycle fromthe given graph G. Figure 3. 2: (a) The truth table for the clause (xWyWz) shows that for theinstance where fx = 1, y = 2 and z = 1g evaluates to true. (b) The graph G= (V, E), where V = f1, 2, 3, 4g and E = f(3, 2),(2, 4),(3, 4),(4, 1),(1, 3)g shows thatfor the instance f3, 2, 4, 1, 3g shows we have a cycle for this instance

- The following de_nition of NP shows that any problem that is in P is alsoin NP. Since if a decision problem is in the decision class P, it indicates that wecan solve it in polynomial time without even being given a certi_cate. 183. 4 NP-completeDe_nition 3. 4. 1 The process of devising an NP-completeness proof for a de-cision problem L will consist of the following four steps: 1. showing that L is in NP, 2. selecting a known NP-complete problem L', 3. constructing a transformation f from L' to L, and4. proving that f is a (polynomial) transformation [14]. In other words, a decision problem L is said to be in the class NP-completeif there is a known NP-complete that can be reduced to the decision problem Lusing a polynomial-time algorithm. Example 3. 4. 2 Some examples of decision problems that fall into the classNP-complete: 1. Vertex Cover - Given a graph G = (V, E) and a positive integer K _ jVj -Is there a vertex cover of size K for G, that is, a subset V' _ V such thatjV'j _ K and, for each edge fu, vg 2 E, at least one of u and v belongs toV'. 2. Clique - Given a graph G = (V, E) and a positive integer J _ jVj - DoesG contain a clique of size J, that is, a subset V' _ V such that jV'j _ Jand every two vertices in V' are joined by an edge in E. Figure 3. 3: Undirected graphs. (a) A undirected graph G = (V, E), whereV = f1, 2, 3, 4, 5, 6, 7g and E = f(7, 1),(1, 2),(2, 3),(2, 6),(3, 6),(3, 5),(6, 5),(3, 4)g wehave a vertex cover V' = f1, 6, 3g such that all the edges in G are covered.(b) A undirected graph G = (V, E), where V = f1, 2, 3, 4, 5, 6g and E = f(1, 2),(2, 3),(3, 4),(4, 5),(3, 5),(5, 6),(6, 2)g we have a clique V' = f3, 4, 5g such thateach pair of vertices is connected by an edge. 19De_nition 3. 4. 3 We say that a decision problem L1 is polynomial-time re-ducible to a problem L2, written L1 _p L2, if there exits a polynomial-timecomputable function f : f0, 1g* ! f0, 1g* such that for all x 2 f0, 1g*, x 2 L1 ifand only if f(x) 2 L1 [24]. In other words, the general notion of a polynomial-time reduction is, givena decision problem L it can be reduced to a problem L' if it's the case that, aninstance of L can be transformed using a polynomial-time to an instance of L', in which the transformation also gives a solution to the instance of L. Example 3. 4. 4 Some examples of decision problems that have been reduced toother decision problems: 1. Clique _p Vertex-cover2. Vertex Cover _p Hamiltonian-cycle3. 3-SAT _p CliqueFigure 3.

4: Undirected graphs. (a) A undirected graph G = (V, E), where V = fu, v, z, w, y, xg and E = f(z, u),(u, v),(v, x),(x, y),(y, u),(y, v),(y, w),(u, x),(z, x),(z, w)ghas the clique V' = fx, u, v, yg. (b The graph G' is constructed by a polynomialtime reduction algorithm, which has a Vertex Cover V - V' = fz, wg. (c) Thegraph G with a clique is constructed from the 3-CNF formulaH= (x1WW : x2: x3)V(: x1Wx2Wx3)V(x1Wx2Wx3) using a polynomual reuctionalgorithm. Lemma 3. 4. 5 If L1 _p L2 then L2 2 P implies L1 2 P. Proof Let A2 be a polynomial-time algorithm that decides L2, and let F bea polynomial-time reduction algorithm that computes the reduction function f. We shall construct a polynomial time algorithm A1 that decides L1 [25]. 20Figure 3. 5: From the diagram we see that F is reduction algorithm that computesthe reduction f from L1 to L2 in polynomial time. A2 is a polynomial-timealgorithm that decides L2. Image from [30]. 21Chapter 4The 8 basic NP-completeproblems4. 1 OverviewIn chapter 4, we present 8 basic NP-complete by providing precise de_nitions. We also take an insight into the history behind all 8 problems. Finally, we goonto provide examples how these problems are modelled in real-time applications. 4. 2 Graph ColouringDe_nition 4. 2. 1 Consider a graph G (recall De_nition 2. 1. 1). A colouringof G with colour-set C is a map f : V (G) ! C such that for each fu; vg 2 E(G)we have f(u) 6= f(v). Such a colouring is called a k-colouring for some integerk _ 0 if jCj _ k. The default colour-set of a k-colouring is f1; : : : ; kg. In other words, a graph colouring uses a colour-set C, and assigns to everyvertex a colour", that is, an element of C, such that adjacent vertices getdi_erent colours. There are many di_erent interpretations of the graph colouring problem. A possible interpretation would be, edge colouring which assigns a colour toevery edge in the graph G, such that no two adjacent edges share the samecolour. Another possible interpretation would be, region colouring whichassigns a colour to each region in the graph G, such that no two regions thatshare the same boundary have the same colour. The graph colouring problem originates from the problem of colouring thecountries of a map such that no two countries that have a common border receivethe same colour. It is possible to transform a map to a planar graph G. Suchthat, every country gets a point in the plane and connect each pair of pointsthat match the countries with a common border by a curve. Then we determinewhether or not every planar can be coloured with 4 colours. 22Figure 4. 1: image from [29]. Example 4. 2. 2 Here are some examples of applications that model the graphcolouring problem: 1. Scheduling - Scheduling an exam timetable can be scheduled in any order, but pairs of exams may cause major problem if they are assigned to thesame time slot. The following graph G would contain a vertex for everyexam and an edge for every conicting pair of exams. 2. Register allocation - Each colour represents an available register. Thefollowing graph G would contain a vertex for each variable if its the casethat variable a and b are live at the same point they cannot be assigned tothe same register. We add an edge (a, b) to the graph. Figure 4. 2: The diagram shows the three exams History, Maths and P. E whichcoincide at the same time on the timetable. English and P. E obviously don'tcoincide with any other subject4. 3 CliqueDe_nition 4. 3. 1 Consider a graph G (recall De_nition 2. 1. 1). A clique of Gis a subset V' _ V of vertices such that every two vertices are connected by anedge in E. 23In other words, a clique determines a complete sub-graph of G, that is, asubset S of vertices such that every two vertices in S are connected by an edgein G. There are many di_erent interpretations of the clique problem. A possibleinterpretation would be, the maximum clique in a given graph G, which isthe sub-graph with the largest possible number of vertices. Another possibleinterpretation would be, the maximal clique, which refers to a sub-graph inwhich no more vertices can be added. Figure 4. 3: A undirected graph G = (V, E), where V = f1, 2, 3, 4, 5, 6g and E= f(1, 2),(2, 3),(3, 4),(3, 5),(5, 6),(6, 2),(6, 1)g with a maximum clique V' = f1, 2, 3gand four maximal cliques = f(2, 3),(3, 4),(3, 5),(5, 6)gThe clique problem started from sociology and psychology as complete cliquewere modelled in terms of social cliques, for instance groups of people that havesome sort of relationship with one another. Example 4. 3. 2 Here is an example of an application that models the cliqueproblem: 1. Social-networks - The maximum clique problem is modelled in social -networks, where the vertices of G would represent people and the edgeswould represent people who are mutual friends. 24Figure 4. 4: In the example above we see that the clique of size 3 (such that the3 people in the diagram know every other in the clique). The three people onthe outside don't know anyone else in the group. 4. 4 Vertex CoverDe_nition 4. 4. 1 Consider a graph G (recall De_nition 2. 1. 1). A vertex coverof G is a subset V' _ such that if (u, v) 2 E, then u 2 V' or v 2 V' (or both). That is, each vertex " covers" its incident edges, and a vertex cover for G is aset of vertices that covers all the edges in E. There is an extension of the vertex cover problem which is referred to as theminimum vertex cover, which determines the minimum number of verticesthat include all the edges in a given graph G. The vertex cover has the followingproperties that, a given graph G is a vertex cover if and only if its complementis an independent set. Also, the vertex cover and the maximum independentset is equal to the jVj. Figure 4. 5: j25Example 4. 4. 2 Here are some simple applications that model Vertex Cover: 1. Computer Network Security - The vertex cover problem has been used inComputer Science to protect computer networks from virus attacks. Thiswas well presented by Eric Filiol [10]. The aim is to _nd a minimum vertexcover, where the vertices are the servers and the edges are the connectionsbetween servers. 4. 5 Hamiltonian CircuitDe_nition 4. 5. 1 Consider a graph G 2. 1. 1). A Hamiltonian cycle of G is an2 E and fvi, vi+1g 2 E for all i, 1 6 i < n. The Hamiltonian cycle problem involves the process of determining whetheror not a given graph G has a edge edge between each pair of consecutive verticesand between the _rst and last vertex. The Hamiltonian cycle problem has thefollowing properties, suppose we have a Hamiltonian cycle H for a given graphG, each vertex in the cycle has precisely two incident edges, one entering thevertex and one leaving. Also, the only Hamiltonian cycle in G is H. Figure 4. 6: jjIn the 1950s, Sir William Hamiltonian a Irish mathematician introducedhis Icosian game at a meeting in Dublin in the late 1950s. The game was to26use a regular dodecahedron whose twenty vertices labelled with names of cities. The aim was to travel around the dodecahedron by _nding a cycle that passesthrough every city exactly once. Example 4. 5. 2 Here are some simple applications that model Hamilto-nian cycles: A Business man - A business traveller leaves every morning from his homewhich is represented by a vertex 1 and needs to visit a number of his clients, which is represented by v2,.., v11 and the edges being all the possible routes thebusinessman can take before returning to his home. How would the businessmanminimise the total distance he travels on his visits?? here we are looking for theHamiltonian cycle for the minimum possible length. 4. 6 Subset-sumDe_nition 4. 6. 1 Given a _nite set S (recall De_nition 2. 2. 1). The subset-sumproblem is a pair (S, t) where S = fx1, x2, xng is a set of positive intgers and t isa postive integer, we ask whether there is a subset S' _ whose elements sum tot. Example 4. 6. 2 He are some applications that use the Subset problem: 1. knapsack - For the knapsack problem we use di_erent variations of a _-nite number of variants to _nd the best possible outcome to a particularsituation, a simple example of this would be as follows given the set S= f20, 30, 10, 2, 5, 25, 41, 3, 15, 3, 1g what is the best possible way of getting avalue t = 40 speci_c to the constraint. 4. 7 3-SATDe_nition 4. 7. 1 Consider a conjunctive normal form formulaH(recall Def-inition 2. 3. 1). A 3-SAT formula consists of a collection C = fc1, c2,..., cmg, where each clause has 3 distinct literals, Ci = li1Wli2Wli3 on a _nite set U ofvariables. Is there a truth assignment for U that satis_es all the clauses in C. The 3-CNF decision problem is the process of determining that there existsan assignment, which satis_es a Boolean Logic formula which is in Conjunctive27Normal Form and has exactly 3 distinct literals in each clause. The 3-CNFproblem wants to establish that the given formula is satis_able such that theformula to TRUE for a given instance. 4. 8 SudokuDe_nition 4. 8. 1 Consider a grid G. A Sudoku S = n2 * n2 grid, which isdivided into n * n sub-grids, such that, each row, column and n * n sub-grid haseach of the integers from 1 through n2 exactly once. The Sudoku problem involves the process of _lling in a given n2 x n2 gridS, such that every row, every column and n x n subgrid has each number atmost once. The Sudoku puzzle game is often represented by a 32 x 32 grid whichconsists of 3 x 3 subgrids, where somes of the boxes are _lled with numbers from1 through to 9 and there are also blanks that need to be _lled, such that all theconstraints of Sudoku are adhered to. The Sudoku problem has the followingtwo properties, that Sudoku problems have unique solutions and that Sudokuproblems can only be solved with only reasoning. The Sudoku puzzle game is a very well known puzzle game that has achievedinternational popularity in recent years. The Sudoku puzzle game was introducedin Japan in the mid 1980's by Nikoli in the paper " Monthly Nikolist". The Sudoku puzzle game appeared on British shores in November of 2004 whenit appeared on the British newspaper, The Times. 4. 9 Latin squareDe_nition 4. 9. 1 Consider a grid G. A Latin Square L = n * n grid, suchthat each row and column has each of the integers from 1 through n exactly once. The Latin square problem involves the process of _lling in a given n x n gridL, such that every row and every column has each number at most once. TheLatin square puzzle is often partially completed, there are also blanks that needto be _lled in, such that all the constraints of Sudoku are adhered to. In 1779, Leonhard Euler a Swiss Mathematician introduced the systematicdevelopment of Latin square, when he posed " The problem of the 36 o_cers". The problem was to arrange 36 o_cers, each having one of six di_erent regiments, in a 6 x 6 square, so that each row and each column captured one o_cerof each rank and just one from each regiment. Subsequently, in the 20th century Arthur Cayley a Canadian Mathematiciandeveloped on the ground work of Leonhard Euler by showing that the multiplicationtable of a group is an appropriately bordered special

Latin square. Example 4. 9. 2 Here are some examples of applications that model Latin squares: 1. Sudoku - The Sudoku puzzle game that grown considerably in popularity inrecent years is based on Latin squares, such that each row and column haseach of the integers from 1 through to n exactly onc. The only additionalconstraint is that the n x n subgrid of a n2 x n2 contains each of thenumbers from 1 through to n2 at most once. 282. KenKen - The KenKen puzzle that has also grown in popularity in recentyears is also based on Latin squares, such that each row and column haseach of the integers from 1 through to n exactly once. The only additionalconstraint is that, each bold-outlined group of boxes is a grouping that con-tains integers which achieve the output result using addition, subtraction, multiplication and division. 29Chapter 5De_nition Of 8 Problems5. 1 OverviewIn chapter 9, for every of the 8 problems, we begin by giving the original sourcewhere its NP-completeness was shown. We present 8 basic NP-complete decisionproblems by precisely de_ning what the decision problem is. This will bepresented by giving the input of the given decision problem, the output of thedecision problem and the size of the problem. 5. 2 Graph Colouring5. 2. 1 DiscoveryThe K-Colourability was proved to be in the complexity class NP-complete inRichard Karp's paper " Reducibility among Combinatorial problems" [19]. Itwas shown that, when K = 2 it can be solved in polynomial time, but remainsNP-complete for all _xed K _ 3. Karp used the 3SAT to be a known NPcompleteproblem and proved a polynomial-time reduction to K-Colourabilityfrom 3SAT. De_nition 5. 2. 1 Consider a graph G (recall De_nition 2. 1. 1) and a positiveinteger K _ jVj. If G is K-colourable then there exists a function f: V ! f1, 2,..., Kg such that f(u) 6= f(v) whenever fu, vg 2 E. In other words, the graph colouring decision problem inputs a graph G anda positive integer K, and outputs a YES if G admits a proper vertex colouringwith K colours, NO otherwise. Example 5. 2. 2 Some simple examples for graph colourings and graph-non-colourings [31]1. The Graph (2, 1) is 2 colourable. 2. The Graph (3, 3) is 3 colourable. 3. The Graph (4, 6) is 4 colourable. 4. The Graph (3, 3) is not 2 colourable. 305. The Graph (4, 6) is not 3 colourable. There are two fundamental steps with the graph colouring process. The _rstis the decision process which we input a graph G with n vertices and choose aninteger K where the graph colouring problem decision in yes. The example ofthe _rst case where the decision would be NP-complete would be the 3 colouringas shown above. The output of the decision would be whether or not the graph G admits aproper vertex colouring with K colours. There is also a special case where agraph is 2 colourable and can be be done in P (Polynomial-time) and this iswere the particular graph that we're colouring is bipartite which is a particulargraph whose vertices can be divided into two disjoint sets were every edge thatconnects a vertex in U to one in V. De_nition 5. 2. 3 The chromatic number of a graph G, denoted by _(G), isthe smallest integer k _ 0 such that G has a k-colouring. Example 5. 2. 4 The three most trivial examples: 1. _((;; ;)) = 0; in general we have _(G) = 0 if and only V (G) = ;. 2. _((f1g; ;)) = 1; in general we have _(G) = 1 if and only if E(G) = ; andV (G) 6= ;. 3. _((f2g; f1g)) = 2; in general we have _(G) = 2 if and only if E(G) = 1and V(G) = 2. 31In other words, The chromatic number refers to the smallest number ofcolours needed to colour a graph G is called the chromatic number and is oftendenoted by _(G). With the optimisation process we will have an input of agraph G with n vertices. We will _rstly have a look at the simplest form whichis not bipartite. We have looked at this previously which is the triangle K3. Example 5. 2. 5 Some more Chromatic number examples: 1. _((f3g; f3g)) = 3; in general we have _(G) = 3 if and only if E(G) = 3and V(G) = 3. 2. _((f4g; f6g)) = 4; in general we have _(G) = 4 if and only if E(G) = 6and V(G) = 4. Example 5. 2. 6 On the other hand the following graphs will not be n-1 colourable: 1. _((f3g; f3g)) 6= 2; if and only if E(G) = 3. 2. _((f4g; f6g)) 6= 3; if and only if E(G) = 6. 32Clearly shown above for the base case if both the sub clauses are true thenas a result the graph-colouring problem is satis_edLemma 5. 2. 7 There is a polynomial-time reduction (recall De_nition 3. 4. 3)from 3SAT (recall De_nition 5. 3. 1) to 3COL (see De_nition 5. 2. 1). Let Ufu1, u2,..., ung and C = fc1, c2,..., cmg be any instance of 3SAT. We must con-truct a graph G = (V, E) such that G is 3COL if and only if 3SAT is satis_able. To prove 3-colourable is NP-complete we use a gadget construction from3SAT by mapping a given 3CNF formulaHto the graph G that consists of avertex for each variable, a vertex for the negation of each variable, 5 vertices foreach clause and 3 special vertices: TRUE, FALSE and BLUE. The edges of thegraph are of two types: literal edges which are independent of the clause andclause edges that depend on the clause. The TRUE and FALSE of the special vertices that are connected by an edgewhich indicates they must be given di_erent colours in the 3 - colouring problem, most often than not True is given the colour green and False is given the colourred. The third special vertex which is BLUE is connected to both the TRUEand FALSE vertices. The 3 special vertices would look as follows below: 33For each clause the variable x is a pair of vertices which associated with twoliterals x and : x. For each clause we have three of these vertices and they are allconnected to the vertex blue by an edge. Such that, the negation : x would havedi_erent colour to x and at least one of the literal vertices must be assigned toTrue. The connecting of variables to the special vertices would look as followsfor the following formula (xWyWz) where x = 1, y = 1 and z = 0: For each clause we will end up with 5 special vertices with 5 edges whichwould be connected in a similar way to the diagram below were the bottomvertex of the triangle is coloured Green or Red according to the outcome of thesub clause, for example for our clause (xWyWz), where hx = 1, y = 1 and z= 0i the bottom vertex would be coloured Green as shown below. Now we suppose that the 3-SAT formulaHis Satis_able. We will show thisby constructing a truthtable for the 3-SAT formulaH(xWyWz) to show thatwe have a satisfying instance of the formula. The truthtable below shows thatwe have a satisfying instance where hx = 1, y = 1 and z = 0i: 34The gadget has the property that it is possible to colour the terminals withany combination of the colours True and False, except for colouring all the literalvertices with False. Such that, in any legal 3-colouring of graph G, if no literalsis coloured Blue, then at least one of these literals is coloured True. For each ground vertex there exist an edge which is connected to True and/orFalse of the master vertex. So using the case of the truth table showing that allthe ground vertices cannot to False, thus the failing of the truth table impliesthe graph cannot be satis_ed. We have stated earlier that at least one of theliterals has to be True and the truth table connects the literal and the mastervertex. 35I will now go onto show a 3 colouring gadget corresponding to the following3-CNF formula: (x1Wx2Wx3)V(: x1W:

x2Wx3)V(x1W: x2Wx3)36These two gadget graphs above show the two cases, when the truth tableis not satis_able, implying that the graph is not satis_able in the _rst casebecause the truth table is not satis_able we _nd that the 3 graph colouring isnot satis_able. The instance when x1 = 1 , x2 = 1 and x3 = 0: This casedoesn't satisfy in the truth table and it's also the case that is doesn't satisfy inthe 3-graph colouring. Secondly, the instance when x1 = 1 , x2 = 0 and x3 = 1: This case in the truth table is satis_ed and it's also the case that the 3-graphcolouring is also satis_able. 375. 3 3-SATDiscoveryThe 3SAT problem was the second problem that was proved to be in the complexityclass NP-complete in Stephen Cook's paper " The complexity of theoremproving procedures" [7]. 2SAT e_eciently solved in polynomial time is shownin [2], but it reamins NP-complete for 3SAT. Stephen Cook used the Satis_abilityproblem to prove a polynomial time reduction to 3SAT. De_nition 5. 3. 1 Consider a conjunctive normal form formulaH(recall De_-nition 2. 3. 1). A 3-SAT decision problem formula consists of a collection C = fc1, c2,..., cmg, where each clause has 3 distinct literals, Ci = li1Wli2Wli3 ona _nite set U of variables. Is there a truth assignment for U that satis_es allthe clauses in C. In other words, the 3SAT decision problem inputs a 3SAT formulaHand athe output is YES ifHhas a satisfying truth assignment for C, NO otherwise. Example 5. 3. 2 Some simple examples of 3-SAT formulas: 1. H= (x1Wx2Wx3)2. H= (x1W: x1W: x2)V(x3Wx3Wx4)V(: x1W: x3W: x4)3. H= (x1W: x2W: x3)V(: x1Wx2Wx3)V(x1Wx2Wx3)Lemma 5. 3. 3 There is a polynomial-time reduction from SAT (see De_nitionXXX) to 3-SAT (see De_nition XXX). Firstly, we construct a binary parse tree for the given SAT formulaH, werethe connectives are nodes and the literals are leaves. For the following SATformulaH= ((x1 $ x2)W:((: x1 ! x3)Wx4))V: x2. The binary parse treewould look as follows: 38The reduction from SAT to binary parse tree show that the literal nodemust have one or two leafs, also we introduce a variable yi for the output ofthe internal node. We then go onto rewrite the original formulaHas the andof the root variable and the conjunction of clauses describing the operation ofeach node. For our SAT formula above the result would look as follows: From the following operation the formulaHobtains a conjunction of theclauses which at most has 3 literals. The _nal requirement is that each clausemust be a disjunction of literals. Secondly, we must convert each of the clauses into disjunctive normal form. This is converted by constructing a truth table for each of the clauses inH

'.

We determine the enteries of the truth table that evaluate to 0 (false). For theclauseH'6 the truth table would look as follows: We build the formula in Disjunctice normal form which is those entries thathave evaluated to 0. So going top down in our truth table forH'6, case 4, 5, 6and 7 would be the entries that are transformed to DNF as they all evaluate to0. We do the trasnformation by looking at the assignment of variables in thecases, and if the variable is assigned to 1 we write down the positive lieral in theclause but if the variable is asssigned to 0 we write the negation of that literal. The Disjunctive Normal Form that is equivalent to : H'6 is as follows:

:

H'6 = (y6V: x1V: x3)W(: y6Vx1Vx3)W(: y6Vx1V: x3)WV (: y6: x1Vx3)39We then go onto convert the formula : H'6 into Conjunctive normal form byusing the DeMorgan's Laws. This operation will complement all the lieteralsand change disjunctions to conjunctions and conjunctions to disjunctions. Thefollowing Disjunctive Normal Form formulaH'6 in Conjunctive Normal Formwould look as follows: H" 6 = (: y6Wx1Wx3)V(y6W: x1W: x3)V(y6W: x1Wx3)V(y6Wx1W: x3)We can claim that the following clauseH" 6 clause in Conjunctive NormalForm is equivalent to the original SAT clauseH'6 as shown below with the followtwo truthtables: We do the following transformation for each clause inH' so that each clause isin Conjunctive Normal Form. The _nal step of the transformation is determingthat each clause has exatctly three distinct literals. Firstly, if the given clause Ci has exactly 3 literals this means that Ci satis_esall the requiremts of 3-CNF and can be included in the formula. Subsequently, if the given clause Ci = (l1Wl2) has 2 distinct literals, weinclude an additional literal to to the clause p and : p as follows (l1Wl2Wp)V(l1Wl2W: p) the following clause is equivalent to (l1Wl2) whether or not p = 0 or p = 1. All the additional literal do is ful_l the requirement of 3-CNF thateach clause must have 3 distinct literals. For instance the clause C4 = (: y4W: y5)V(y4Wy5) from our formulaH" is very much equivalent to (: y4WW : y5p)V(: y4W: y5W: p)V(y4Wy5Wp)V(y4Wy5W: p) as shown below: 40Finally, if the given clause Ci = (l1) has 1 distinct literal, we include additionalliterals to the clause p and q as follows: (l1WpWq)V(l1WpWV : q)(l1W: pWq)V(l1W: pW: q). The following clause is equivalent to l1whether or not p = 1 or p = 0 and q = 1 and q = 0. Again, all the additionalliteral do is ful_l the requirement that each clause must have 3-distinct literls. For instance the clause Ci = (xi), is very much equivalent to (xiWpWq)V(xiWpW: q)V(xiW: pWq)