COT 6410 Spring2014 Final Exam Sample E2 Key

1. We described the proof that **3SAT** is polynomial reducible to Subset-Sum.

   a.) Describe **Subset-Sum**

   Let $n_1, n_2, \ldots, n_k$, G be a set of k positive whole numbers and G be a goal number. The decision problem is to determine if there is a subset $n_{i1}, n_{i2}, .., n_{ij}$ of the original set that sums to G.

   b.) Show that **Subset-Sum** is in **NP**

   Let $n_1, n_2, \ldots, n_k$, G be an instance of SubsetSum and let $n_{i1}, n_{i2}, .., n_{ij}$ be a proposed subset. We merely need to add these j numbers together and check that they sum to G. If so, we verify the proposed solution; else we reject it. That process is linear and hence there is a polynomial time verifier, and so SubsetSum is in NP.

   c.) Assuming a **3SAT** expression **(a + ~b + c) (~a + b + ~c)**, fill in the upper right part of the reduction from **3SAT** to **Subset-Sum**.

	a	**b**	**c**	**a + ~b + c**	**~a + b + ~c**
a	1			1	
~a	1				1
b		1			1
~b		1		1	
c			1	1	
~c			1		1
C1				1	
C1'				1	
C2					1
C2'					1
	1	1	1	3	3

   d.) List some subset of the numbers above (each associated with a row) that sums to **1 1 1 3 3**. Indicate what the related truth values are for **a**, **b** and **c**.

   a = T; b = T; c = T

   a 1 0 0 1 0
   b 0 1 0 0 1
   c 0 0 1 1 0
   C1 0 0 0 1 0
   C2 0 0 0 0 1
   C2' 0 0 0 0 1
   SUM 1 1 1 3 3

2. **Partition** refers to the decision problem as to whether some set of positive integers **S** can be partitioned into two disjoint subsets whose elements have equal sums. **Subset-Sum** refers to the decision problem as to whether there is a subset of some set of positive integers **S** that precisely sums to some goal number **G**.

a.) Show that **Partition ≤p Subset-Sum**.

Look at notes

b.) Show that **Subset-Sum ≤p Partition**.

Look at notes

3. Consider the decision problem asking if there is a coloring of a graph with at most k colors, and the optimization version that asks what is the minimum coloring number of a graph. You can reduce in both directions. So, do that. Make sure you carefully explain for each direction just what it is that you are proving.

1. Show that k-Color is polynomial time Turing reducible to MinColor:
Let G=(V, E) be an arbitrary graph and k>0 an arbitrary positive whole number. We can solve k-Color by asking an oracle for MinColor to provide the minimum coloring of G. We then check the MinColor answer. If it is less than or equal to k, we answer yes; else we answer no. This proves that MinColor is NP-Hard, based on our knowledge that k-Color is NP-Complete.

2. Show that MinColor is polynomial time Turing reducible to k-Color:
Let G=(V, E) be an arbitrary graph. Let n=|V|. Do a binary search using an oracle for k-Color in order to find the minimum k such that k-Color returns yes. This takes at most log n questions of the oracle. Thus, MinColor is NP-Easy since we asked just log n questions, and spent just polynomial time at each stage of this Turing reduction. Together, these show MinColor is in NP-Equivalent.

4. **QSAT** is the decision problem to determine if an arbitrary fully quantified Boolean expression is true. Note: **SAT** only uses existential, whereas **QSAT** can have universal qualifiers as well so it includes checking for Tautologies as well as testing Satisfiability. What can you say about the complexity of **QSAT** (is it in **P**, **NP**, **NP-Complete**, **NP-Hard**)? Justify your conclusion.

QSAT is NP-Hard. This is so since SAT trivially reduces to QSAT (it is a subproblem of QSAT). Since SAT is known to be NP-Complete then some NP-Complete problem polynomially reduces to QSAT. This makes QSAT NP-Hard. As we cannot (at least not yet) show QSAT is in NP, then NP-Hard is the best we can do.

5. Consider the following set of independent tasks with associated task times:
(T1,7), (T2,6), (T3,2), (T4,5), (T5,6), (T7,1), (T8,2)
Fill in the schedules for these tasks under the associated strategies below.

Greedy using the list order above:

T1	T1	T1	T1	T1	T1	T1	T4	T4	T4	T4	T4	T7	T8	T8
T2	T2	T2	T2	T2	T2	T3	T3	T5	T5	T5	T5	T5	T5	

Greedy using a reordering of the list so that longest running tasks appear earliest in the list:

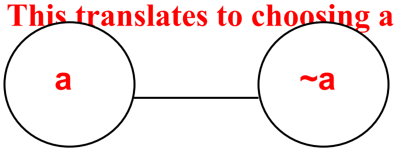
T1	T1	T1	T1	T1	T1	T1	T4	T4	T4	T4	T4	T3	T3	T7
T2	T2	T2	T2	T2	T2	T5	T5	T5	T5	T5	T5	T8	T8	

Greedy using a reordering of the list so that shortest running tasks appear earliest in the list:

T7	T8	T8	T2	T2	T2	T2	T2	T2	T2	T1	T1	T1	T1	T1	T1	T1
T3	T3	T4	T4	T4	T4	T4	T5	T5	T5	T5	T5	T5				

6. Present a gadget used in the reduction of **3-SAT** to some graph theoretic problem where the gadget guarantees that each variable is assigned either **True** or **False**, but not both. Of course, you must tell me what graph theoretic problem is being shown **NP-Complete** and you must explain why the gadget works.

Vertex Cover
Must Cover each Edge
Set goal to min vertices
Must choose one but not both are needed
This translates to choosing a or ~a



3-Color
Cannot choose B for either a or ~a
So one must be T and other F