

Quiz 3

April 27, 2005

Vinod Vaikuntanathan

Please write your name in the upper corner of each page.

Problem	Points	Grade
1	20	
2	10	
3	10	
4	20	
5	20	
6	20	
Total	100	

Name: _____

Problem 1: True, False, or Unknown (20 points). In each case, say whether the given statement is known to be TRUE, known to be FALSE, or currently not known either way. Full credit will be given for correct answers. If you include justification for your answers, you may obtain partial credit for incorrect answers.

1. True, False, or Unknown: 8^n is $O(2^n)$.

False.

2. True, False, or Unknown: There exists a language that is not decidable in time n^8 but is decidable in time 8^n (where n is the length of the input).

True. Invoke the Time-Hierarchy Theorem.

3. True, False, or Unknown: $\text{SAT} \in \text{coNP}$.

Unknown. $\text{SAT} \in \text{coNP}$ if and only if $\text{NP} = \text{coNP}$, which is not known.

4. True, False, or Unknown: The 2COLOR problem is NP-complete.

Unknown. $2\text{COLOR} \in P$. Therefore, it is NP-complete if and only if $P = \text{NP}$, which is not known.

Name: _____

5. True, False, or Unknown: The 4-CLIQUE problem, defined as the set of undirected graphs containing a 4-CLIQUE, is in P.

True. To decide if a graph has a 4-clique, check all 4-tuples of nodes to see if any of them form a clique in the graph. If any of them form a clique, accept. Else, reject.

6. True, False, or Unknown: 2SAT (the set of CNF Boolean formulas with at most two literals per clause) \leq_p PALINDROMES.

True. $2SAT \in P$. Therefore, it trivially reduces to any language $L \neq \Sigma^*, \phi$.

7. True, False, or Unknown: There is a nontrivial language A such that $P^A \neq NP^A$.

True. Invoke Theorem 9.19 of Sipser.

8. Suppose there is a 2^n time-bounded reduction from language L_1 to language L_2 , and L_2 is in $\text{TIME}(2^n)$. Then, L_1 is in $\text{TIME}(2^{O(n)})$.

False. By Time Hierarchy Theorem, there is a language L_1 in $\text{DTIME}(2^{2^n})$ but not in $\text{DTIME}(2^{1.5^n})$. Define $L_2 \stackrel{\text{def}}{=} \{xy \mid |y| = 2^{|x|}, x \in L_1\}$. Now, L_2 is in $\text{DTIME}(2^n)$. Why? A decider for L_2 looks at the first $\log n$ bits of an n -bit input and decides if the $\log n$ -bit string is in L_1 , which can be done in time $2^{2^{\log n}} = 2^n$.
 L_1 reduces to L_2 in time 2^n via a function f that takes x and outputs xy for an arbitrary y of length $2^{|x|}$. But, we know that L_1 is not in $\text{TIME}(2^{O(n)})$.

Name: _____

Problem 2: (10 points) Suppose that A , B , and C are nontrivial languages over $\Sigma = \{0, 1\}$. Assume that:

1. $A \in NP$,
2. $C \in P$,
3. $A \leq_p B \leq_p A \cap C$, and
4. $A \cap C$ is NP-hard.

Prove that B is NP-complete. You may invoke theorems proved in class and in the book, but if you do this, cite them explicitly.

1. $B \in NP$. Why?
 - 1.1. $A \in NP, C \in P$. Given.
 - 1.2. $C \in NP$ since $NP \supseteq P$.
 - 1.3. Since NP is closed under intersection, $A \cap C \in NP$.
 - 1.4. $B \leq_p A \cap C$. Given.
 - 1.5. From 1.3 and 1.4, $B \in NP$. ■
2. B is NP-hard.
 - 2.1. Let $D \in NP$.
 - 2.2. $D \leq A \cap C$, since $A \cap C$ is NP-hard.
 - 2.3. $A \cap C \leq A$ (This is a homework problem).
 - 2.4. $D \leq A$. Transitivity.
 - 2.5. $A \leq B$. Given.
 - 2.6. $D \leq B$. Transitivity.
 - 2.7. Therefore, for all $D \in NP$, $D \leq_p B$. ■

Name: _____

Problem 3: (10 points) The proof that SAT is NP-complete appears in Section 7.4 of Sipser's book.

1. Suppose we try to use exactly the same construction to show that every problem A in the class NEXPTIME, representing nondeterministic exponential time (time $O(2^{n^k})$ for some k), is polynomial time reducible to SAT. Exactly where would the proof fail?

The tableau created would be of size $2^{n^k} \times 2^{n^k}$. Since we had a variable for each cell of the tableau, the formula would have to be of size 2^{2n^k} . The reduction, thus, cannot run in polynomial time.

2. The overall formula ϕ actually depends on the input x for the given nondeterministic Turing machine. Explain in words where the input appears in the formula.

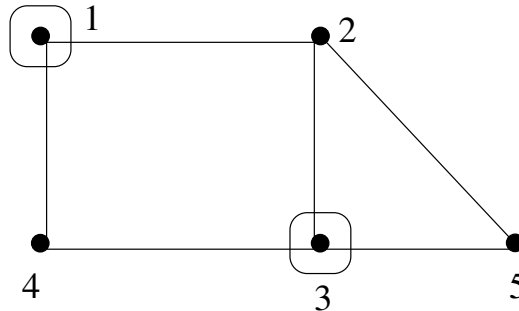
The input x appears in the sub-formula ϕ_{init} , which expresses the condition that the first row of the tableau corresponds to the initial configuration of the TM. The initial configuration is one in which the tape contains only the input.

Name: _____

Problem 4: (20 points) The following is a variation on the Vertex Cover problem, called Dominating Set:

$DS = \{ \langle G, k \rangle : G \text{ is an undirected graph } (V, E), \text{ and there exists a subset } C \subseteq V, \text{ with } |C| = k,$
and such that every vertex in $V \setminus C$ has an edge in E to some vertex in $C \}$.

For example, in the following graph G , $C = \{1, 3\}$ is a dominating set of size 2:



1. Prove that DS is in NP, by describing a polynomial-time nondeterministic Turing machine that decides DS.

The machine M first chooses some subset $C \subseteq V$ non-deterministically.

It then checks that:

- (1) $|C| = k$, and
- (2) Every vertex in $V \setminus C$ has an edge to some vertex in C . This can be done in time $O(|V|^3)$, for example, by looking at all the adjacencies of every node in $V \setminus C$ and checking if at least one of them is in C .

If both checks pass, accept. Else, reject.

Name: _____

2. Prove that DS is NP-hard, using a polynomial time reduction from the Vertex Cover problem.

(Hint: From the given graph $G = (V, E)$, construct a graph $G' = (V', E')$ such that $V' = V \cup E$.)

Given (G, k) for the Vertex-cover problem, where $G = (V, E)$, produce (G', k) for DS, where $G' = (V \cup E, E')$.

So, G' has vertices corresponding to both the vertices and edges of G .

$E' = E \cup \{(u, e) \mid u \in V, e \in E, u \text{ is one of the endpoints of } e\}$

That is to say, add in edges between the edges of G and their incident vertices.

For the forward direction, we prove that a k -vertex cover in G is also a k -dominating set in G' .

It is easy to see that a k -Vertex cover C in G is also a k -dominating set for G . What about the extra vertices in G' ? Consider some such vertex e . Then, since C is a vertex cover in G , we know that at least one of the vertices in C is adjacent to every edge e of G . This, in turn, means that the vertex e in G' is adjacent to one of the vertices in C . Thus, C is a k -dominating set in G' .

For the converse, assume that there is a k -DS C' in G' .

Divide C' into two disjoint sets C'_V and C'_E , where C'_V consists of those vertices that are in G too, and C'_E consists of those vertices that we added (to G to make G').

Now, form a k -VC C for G as follows: Add to C all the vertices in C'_V . For each vertex e in C'_E , add one of the end-points of the edge e in G to C .

How do we see that this is indeed a k -VC for G ? Consider any edge e in G . The corresponding vertex e of G' has to be adjacent to one of the vertices in either C'_V or C'_E (because, $C' = C'_V \cup C'_E$ is a dominating set for G'). If it is adjacent to a vertex in C'_V , we are done.

Otherwise, it is adjacent to a vertex in C'_E . But this latter case is not possible, because no two "new" vertices e and e' are adjacent in G' .

Name: _____

Problem 5: (20 points) Suppose we are given a finite set S of elements, together with a finite collection of subsets $S_1, S_2, S_3, \dots, S_k$ of S . Also, for each set S_i , we are given two numbers low_i and $high_i$. The problem is to figure out whether it is possible to select a subset T of S such that, for every i , the number of elements of S_i in T is in the interval $[low_i, high_i]$. i.e, for all i , $low_i \leq |T \cap S_i| \leq high_i$. For example, consider

$$\begin{aligned} S &= \{a, b, c, d, e\} \\ S_1 &= \{a, b, e\}, low_1 = 1, high_1 = 1 \\ S_2 &= \{a, c, d\}, low_2 = 1, high_2 = 2 \\ S_3 &= \{b, c, d, e\}, low_3 = high_3 = 3 \end{aligned}$$

This instance is in the language. To see this, set $T = \{b, c, d\}$. T intersects S_1 in one element b , S_2 in two elements c and d , and S_3 in three elements b, c and d .

However, the same example, with $low_2 = high_2 = 1$, has no choice for T that works.

1. Define this problem formally as a language called SUBSETS.

SUBSETS = $\{ \langle S, k, \{S_i, low_i, high_i\}_{i=1}^k \rangle \mid S \text{ is a finite set, For all } 1 \leq i \leq k, S_i \subseteq S, \exists T \subseteq S$
such that for all $1 \leq i \leq k, low_i \leq |T \cap S_i| \leq high_i \}$.

2. Prove that SUBSETS is in NP, using the certificate/verifier formulation.

The Certificate: a subset $T \subseteq S$.

The Verifier does the following:

- (1). For each $1 \leq i \leq k$, verify that $low_i \leq |T \cap S_i| \leq high_i$.

This verifier runs in time at most $O(k|S|^2)$, which is clearly polynomial in the input size.

Name: _____

3. Prove that SUBSETS is NP-hard, using a direct reduction from 3SAT.

From a 3cnf formula ϕ with variables $\{x_1, \dots, x_l\}$ and clauses $C_1 \dots C_k$, construct an instance of SUBSETS as follows:

(1) $S = \{x_1, \dots, x_l\} \cup \{\bar{x}_1, \dots, \bar{x}_l\}$. That is, S is the set of all variables along with their complements.

(2) Construct sets S_i ($1 \leq i \leq l + k$) as follows:

For $1 \leq i \leq l$, set $S_i = \{x_i, \bar{x}_i\}$. $low_i = high_i = 1$.

For $l + 1 \leq i \leq l + k$, look at clause C_{i-l} , and construct set S_i to contain the 3 literals in the clause. Also, set $low_i = 1$ and $high_i = 3$.

First of all observe that any certificate $T \subseteq S$ should have *exactly* one element from each set S_i ($1 \leq i \leq l$). This corresponds directly to a satisfying assignment. If $x_i \in T$, set the variable x_i to *true* in the assignment. If $\bar{x}_i \in T$, then set $x_i = false$. This satisfies every clause since $|T \cap S_i| > 0$ (for all $l + 1 \leq i \leq l + k$).

Conversely, if there is a satisfying assignment, each variable x_i is set to either *true* or *false* (not both). This corresponds to a choice of either x_i or \bar{x}_i from S to include in T . More precisely, if $x_i = true$, set $T \leftarrow T \cup \{x_i\}$, else set $T \leftarrow T \cup \{\bar{x}_i\}$. T is easily seen to contain at least one element from the set corresponding to each clause.

Name: _____

Problem 6: (20 points) Consider the language 3DM (Three-Dimensional Matching) defined in class and in Sipser's book. That is,

$3DM = \{\langle A, B, C, M \rangle \mid A, B, \text{ and } C \text{ are disjoint sets of the same size, } M \subseteq A \times B \times C, \text{ and there exists } M' \subseteq M \text{ such that every element of } A, B, \text{ and } C \text{ appears in exactly one triple in } M'\}.$

Suppose that we are given a polynomial-time decider algorithm D that decides membership in 3DM.

1. Describe a polynomial time algorithm that, given the representation $\langle A, B, C, M \rangle$, finds a set $M' \subseteq M$ such that every element of A,B, and C appears in exactly one triple in M' , if one exists, and otherwise outputs "none exists". Your algorithm may use the decider D as a "subroutine".

There are (at least two) different ways to solve this problem. We present one here:

First, check if $\langle A, B, C, M \rangle \in 3DM$ using D . If not, output \perp (it is hopeless to try to find an M').

If yes, first set $M' = \phi$. Then,

1. Remove a triple $\langle a, b, c \rangle$ from M (that has not been removed before).

i.e, set $M'' = M - \{\langle a, b, c \rangle\}$.

2. Now run D on $\langle A, B, C, M'' \rangle$.

2.1. If D accepts, we know that we can find a good matching M' as a subset of M'' . So, throw out $\langle a, b, c \rangle$. Set $M := M''$ and go to Step 1.

2.2. If D rejects, we know that $\langle a, b, c \rangle$ is essential for M' . So, set $M := M'' \cup \{\langle a, b, c \rangle\}$ and $M' := M' \cup \{\langle a, b, c \rangle\}$. Goto step 1.

The proof that this algorithm works, is left as an exercise.

Name: _____

2. Show that your algorithm in part 1 is actually polynomial time-bounded. Specifically, assuming that the decider D runs in time $p(n)$ for polynomial p (of degree at least 2), give a specific polynomial time upper bound for your algorithm (including the time for the “subroutine” D).

Suppose the size of the input is n , and suppose the running time of D on inputs of size n is $T_D(n)$. Then, since D is run at most once on each triple, the total time taken is $O(nT_D(n))$.