HW 13 CMSC 452. Morally Due May 8
SOLUTIONS

1. (5 points) What is your name? Write it clearly. Staple the HW.

2. (30 points) Let f be a function. The *image of* f is the set of all y such that there is some x where $f(x) = y$. Formally

$$\text{image}(f) = \{y : (\exists x \in A)[f(x) = y]\}$$

For each of the following either say TRUE or FALSE or UNKNOWN TO SCIENCE. If TRUE then prove it, if FALSE then you **do not** have to prove it, if UNKNOWN TO SCIENCE, you don't have to resolve it.

(a) Let f be a computable function such that

$$(\forall x, y)[x < y \rightarrow f(x) < f(y)]$$

Then the image of f is computable.

(b) Let f be a computable function such that

$$(\forall x, y)[x < y \rightarrow f(x) \le f(y)]$$

Then the image of f is computable.

(c) Let f be computable in polynomial time. Then the image of f is in P

SOLUTION TO PROBLEM TWO

a) TRUE.

Algorithm for the image of f: On input y compute

$f(1)$, $f(2)$, $f(3)$, ...

until EITHER

- You find an x such that $f(x) = y$. Then output YES.
- You find an x such that $f(x) > y$. Then output NO. Since f is increasing there will be no $x' > x$ with $f(x') = y$, and since you didn't do step 1 there was no $x' \le x$ with $f(x') = y$.

b) TRUE

There are two cases. KEY- you DO NOT HAVE TO know which case you are in, just that you must be in one of them.

CASE ONE: $f(x)$ goes to infinity. So it may go slowly, it may be that $f(1) = f(2) = \cdots f(100000) = 1$ and then you get 2 for a long time, etc. but you eventually go to infinity. Then do the algorithm in the last part.

CASE TWO: $f(x)$ is eventually constant. Hence the image of f is finite. All finite sets are computable.

c) UNKNOWN TO SCIENCE. If this was TRUE then consider the following function

$$f(\phi, x) = \begin{cases} \phi & \text{if } \phi(x) = T \\ x & \text{otherwise} \end{cases} \tag{1}$$

This function is in P since it only involves evaluating a formula.

The image is SAT (note that $x \in SAT$).

If $P = NP$ then the image is in P.

One can show that if $P = NP$ then ALL images of polytime computable functions are in P.

END OF SOLUTION TO PROBLEM TWO

3. (35 points) Show that there exists a decidable set that is **not** in $\bigcup_{a=1}^{\infty} DTIME(2^{n^a})$

SOLUTION TO PROBLEM THREE

Let $T(n) = 2^{2^n}$. Note that if $A \in \bigcup_{a=1}^{\infty} \subseteq (2^{n^a})$ then $A \in DTIME(T(n))$.

Take the set of all Turing Machines. To each one attach a counter so that if on input n it goes for $T(n)$ steps then shut it off and output NO. Further modify these machines such that the output is either YES or NO.

$$M_1, M_2, M_3, \ldots$$

We write a program for a set B such that $B \notin DTIME(T(n))$, and hence $B \notin \bigcup_{a=1}^{\infty} \subseteq (2^{n^a})$. It will be a subset of 0^*.

(a) Input(0^e)

(b) Run $M_e(e)$. If it says NO then output YES. If the output is YES then say NO.

We claim that, for all e, M_e DOES NOT decide B.

IF $M_e(0^e) = YES$ then but the construction $0^e \notin B$, hence M_e and B differ on 0^e.

IF $M_e(0^e) = NO$ then but the construction $0^e \in B$, hence M_e and B differ on 0^e.

In either case M_e and B differ on 0^e. Hence M_e does not decide B. Since this argument holds for any e, NO M_e decides B. So $B \notin DTIME(T(n))$ and hence $B \notin \bigcup_{a=1}^{\infty} \subseteq (2^{n^a})$.

END OF SOLUTION TO PROBLEM THREE

4. (30 points) Let $COUNTSAT$ be the **function** that takes a boolean formula and outputs **the number** of satisfying assignments it has. (The answer could be 0.)

   (a) True or False: If $COUNTSAT$ can be computed in polytime, then $P = NP$. In either case justify your answer.

   (b) Write an algorithm for $COUNTSAT$.

   (c) How fast does your algorithm run (express as a function of n, the number of variables).

SOLUTION TO PROBLEM FOUR

a) TRUE:

Assume $COUNTSAT$ can be computed in poly time. Then

$$SAT = \{\phi : COUNTSAT(\phi) \geq 1\}$$

Testing if $COUNTSAT(\phi) \geq 1$ is in Poly, so SAT is in Poly, so $P = NP$.

b) ALGORITHM: on input ϕ evaluate ϕ on ALL 2^n assignments. Keep a counter. Every time an assignment makes ϕ true add to the counter. At the end output the counter.

c) The algorithm goes through ALL 2^n possible assignments. For each one it evaluates which takes around n steps. So the time is $n2^n$.

END OF SOLUTION TO PROBLEM FOUR