

## CS303 (Spring 2008)- Solutions to Assignment 12

### Problem 1

- (a) An efficient certifier is an algorithm/program  $A(x, y)$  with two inputs  $x, y$ , and the following properties:
  - (1) It always runs in time polynomial in the size of  $x$  and  $y$ .
  - (2) If  $x$  is a “Yes” instance for the underlying problem, then  $A(x, y)$  outputs “Yes” for at least one possible input  $y$  of length at most polynomially longer than  $x$ .
  - (3) If  $x$  is a “No” instance for the underlying problem, then  $A(x, y)$  always outputs “No”, regardless of what  $y$  is.
- (b) NP is the class of all decision problems that have efficient certifiers. Alternatively, NP is the class of all decision problems that can be decided by a non-deterministic Turing Machine in polynomial time.
- (c) Informally, NP is the class of problems for which a solution given by someone else can be verified efficiently (in polynomial time).
- (d) We mean that there is a *polynomial-time* reduction from  $X$  to  $Y$ , i.e., a function  $f$  always running in polynomial time with the following properties: (1) If  $x \in X$ , then  $f(x) \in Y$ , and (2) If  $x \notin X$ , then  $f(x) \notin Y$ .
- (e) It means that *every* problem in NP can be reduced in polynomial time to the given problem. That is,  $X$  is NP-hard is  $Y \leq_p X$  for all problems  $Y$  in NP.
- (f) It means that the problem is NP-hard, and also itself in NP.

### Problem 2

If  $X \leq_p Y$  and  $Y \leq_p Z$ , that means that we have functions  $f$  running in polynomial time  $T_f(x) = O(|x|^k)$ , and  $g$  running in polynomial time  $T_g(x) = O(|x|^\ell)$  with the following properties: (1)  $x \in X$  if and only if  $f(x) \in Y$ . (2)  $y \in Y$  if and only if  $g(y) \in Z$ . Now, we simply execute the corresponding programs one after the other. First we run  $f$  on  $x$ , and then we run  $g$  on the output  $f(x)$ . Call the resulting function  $h$ . Because  $f$  takes time  $O(|x|^k)$ , its output can have size at most  $O(|x|^k)$ . So when  $g$  runs on  $f(x)$ , it can take time at most  $O((|x|^k)^\ell) = O(|x|^{k\ell})$ , which is also polynomial. Furthermore, whenever  $x \in X$ , we get  $f(x) \in Y$  by property (1), and so  $g(f(x)) \in Z$  by property (2). Similarly, when  $x \notin X$ , we have  $f(x) \notin Y$ , and so  $g(f(x)) \notin Z$ . Thus,  $h$  is a correct polynomial-time reduction from  $X$  to  $Z$ , and we have proved  $X \leq_p Z$ .

### Problem 3

The algorithm is simply to do an exhaustive search over all possible certificates, and then run the efficient certifier for each of them. If at least one of the results is “Yes”, our algorithm outputs “Yes”, and otherwise “No”.

How many certificates are there to test? If the input is  $x$ , then the definition of an efficient certifier limits certificates to length at most  $p(x)$  for some polynomial  $p$ . If we assume binary strings (which we can without losing anything), then there are  $2^{p(|x|)}$  certificates to test, and each test takes polynomial time, which is less than another factor  $2^{|x|}$ . Therefore, the overall running time is  $O(2^{p(|x|)+|x|})$ , which is itself  $O(2^{p'(|x|)})$  for the polynomial  $p'(x) = p(x) + x$ .

### Problem 4

- (a) This is the TRAVELING SALESMAN problem. The decision version is: Given  $G$  and the  $d_e$  as well as a total length bound  $L$ , is there a tour visiting each vertex at least once, and of total length at most  $L$ ?

To prove that it is in NP, we notice that a certificate is a sequence in which to visit each node. We can then test in polynomial time (e.g., by setting bits in an array) that each node appears in the sequence at least once. Also, we can add up the length of all edges in the sequence (linear time), and compare the sum to  $L$ . If it is at most  $L$ , and all vertices are visited, the certifier accepts; otherwise, it does not.

- (b) This is called MATCHING. It can actually be solved in polynomial time. In general graphs, the algorithm is quite complex. In bipartite graphs (say, matching boys and girls, or jobs and machines), it becomes easier, and can be solved using Maximum Flow (Chapter 26.3 in the textbook).

The decision version is: Given the graph  $G$  and a number  $k$ , are there  $k$  pairs of people with the property that (1) no one is part of more than one pair, and (2) each pair is connected by an edge?

To see that it is in NP, we can use as a certificate a set of edges  $M$  (clearly polynomial in size). The certifier test that (1) each of the edges of  $M$  is really in the graph, and (2) no person is part of more than one edge (by filling in an array of bits or such). If both (1) and (2) are satisfied, the certifier says “Yes”; otherwise, it says “No”. This clearly runs in polynomial time.

- (c) This is another well-known and important problem, called KNAPSACK. It is often at the core of efficient resource usage. The decision version is: Given the weights and values of all items, as well as a weight bound  $W$  and a value bound  $V$ , is there a set  $S$  of items of total weight at most  $\sum_{i \in S} w_i \leq W$  and total value at least  $\sum_{i \in S} v_i \geq V$ ?

The certificate is the set  $S$ , which takes polynomial space to write down. The certifier takes the set  $S$ , and sums up the weights of all elements in  $S$ , comparing the sum to  $W$ . It then sums up all the values, and compares the sum to  $V$ . If the total weight is at most  $W$ , and the total value at least  $V$ , the certifier accepts; otherwise not. This just involves one or two loops, and thus runs in polynomial time.

## Problem 5

Obviously, we can't really give a sample solution for this one.