CS3230: Design and Analysis of Algorithms (Fall 2014)

Tutorial Set #6

[For discussion during Week 8]

S-Problems are due (outside Prof. Leong's office): Friday, 3-Oct, before noon.

OUT: 27-Sep-2014 **Tutorials:** Tue & Wed, 7, 8 Oct 2014

IMPORTANT: Read "Remarks about Homework".

Submit solutions to S-Problem(s) by deadline given above.
Prepare your answers to all the D-Problems in every tutorial set.

When preparing to present your answers,

- Think of a CLEAR EXPLANATION
- Illustrate with a good worked example;
- Describe the main ideas,
- Can you sketch why the solution works;
- Give analysis of running time, if appropriate
- Can you think of other (perhaps simpler) solutions?

Helpful Hints Series: A problem well understood is half solved ☺. Since this topic is quite new and abstract, please become very clear of the definitions. Please read and understand the questions slowly and carefully.

Routine Practice Problems *-- do not turn these in -- but make sure you know how to do them.*

R1. COMPOSITE = Given a natural number k, decide if k is a composite natural number?

Show that COMPOSITE is in NP.

Answer. The solution appears in lecture notes.

R2. Show that deciding whether two graphs G and H are isomorphic is in NP (assume that the graphs are specified using adjacency table.)

Answer. Two graph G= (V1, E1) and H=(V2,E2) are said to be isomorphic if there exists a permutation t of the vertices of G such that for all u $\in$ V1 and v$\in$ V1,

$$(u,v) \in E1 \iff (t(u),t(v)) \in E2 .$$

In order to show that a decision problem L is in NP, we need to exhibit a certifier C which takes as input (x,t) (x and t are binary strings and t is referred to as a certificate) and runs in worst case time bounded by p(|x|), where p is a fixed polynomial (|x| represents the number of bits in x). Following conditions must be satisfied.

- If x $\in$ L, there exists some certificate t, for which C(x,t) = true.
- If x $\notin$ L, then for all certificates t, C(x,t) = false.

In this case the input to certifier C would be (G,H,t) (here x=(G,H)). Here certificate t would be a valid isomorphism between G and H, which can be represented by a permutation of the vertices of G (using some binary encoding). The certifier C is as follows.

```
C(G = (V1, E1), H=(V2,E2), t) {
        If (G and H are not valid binary encodings of graphs)
                Return false
        Else if (t is not a valid permutation of vertices of G)
                Return false
        Else if (V1 ≠ V2)
                Return false
        For all u  ∈ V
                For all v ∈ V
                        If ( (u,v) ∈ E1 and (t(u),t(v)) ∉ E2 )
                                Return false
                        Elseif ( (u,v) ∉ E1 and (t(u),t(v)) ∈ E2 )
                                Return false
        Return true
}
```

Let n = |V1|+ |V2|. The input x=(G,H) is specified using $\Theta(n^2)$ bits (using adjacency table). It is easily seen that the worst case running time of C is bounded by a polynomial in n^2 .

R3. Show that deciding whether a graph G is a subgraph of another graph H is in NP.

Answer. A graph G = (V1, E1) is a sub graph of another graph H=(V2,E2)) if V1 $\subseteq$ V2 and E1 $\subseteq$ E2. This problem is in fact in P. The polynomial time algorithm A is as follows.

```
A(G = (V1, E1), H=(V2,E2)) {
        If (G and H are not valid binary encodings of graphs)
                Return false
        Else if (V1 ⊄ V2)
                Return false
        Else if (E1 ⊄ E2)
                Return false
        Return true }
```

S-Problems: (To do and submit by due date given in page 1)
Solve this S-problem(s) and submit for grading.

IMPORTANT: Write your NAME, Matric No, Tutorial Group in your Answer Sheet.

S1. [Understanding P and polynomial time reductions]

Please recall that decision problems can be viewed as languages which are subsets of {0,1}* (the set of all finite length binary strings).

Let A be a decision problem. Show that if A $\in$ P, then for every decision problem B, we have: A $\leq_P$ B (unless B or complement of B is empty).

Answer.
Let B be a decision problem (a.k.a language) such that neither B nor complement of B ($\overline{B}$) is empty. In order to show that A $\leq_P$ B we need to exhibit a polynomial time algorithm R such that

$$x \in A \Leftrightarrow R(x) \in B \text{ .}$$

Here R(x) represents the output of R on input x. Such R is also known as Karp reduction from A to B.

Let y $\in$ B and z $\notin$ B (y, z exist since neither B nor $\overline{B}$ is empty). Let A also represent the polynomial time algorithm deciding A. The reduction R is as follows.

```
R(x) {
        If A(x) = true
                Return y
        Elseif A(x) = false
                Return z
}
```

It is easily seen that $x \in A \Leftrightarrow R(x) \in B$.

D-Problems: Solve these D-problems and prepare to discuss them in tutorial class. You may be called upon to present your solution *or your best attempt at a solution*. Your solution presentation does NOT need to be fully correct, given your best attempt. The TA will help clarify and correct any issues or errors.

D1. [Self-reducibility]

FACTORIZE. Given an integer x, find its factorization (into nontrivial factors).
Number p is called a nontrivial factor of x if $1 < p < x$ and p divides x.

FACTOR. Given two integers x and y, decide if x has a nontrivial factor less than y?

Show that (using Cook reductions): FACTOR $\equiv_P$ FACTORIZE.

Answer.

When we reduce problem A to problem B using a Cook reduction, we present a polynomial time algorithm R, which uses oracle calls for B, to solve A. The oracle calls to B are counted as unit time.

1. Reducing FACTOR to FACTORIZE

```
R1(x,y) {
        Set {p1,p2, …, pr} =  FACTORIZE(x)  (the factorization of x)
        If (any of {p1,p2, …, pr } is less than y)
                Return true
        Else
                Return false
}
```

The input size is $(lgx + lgy)$. Since $r \le lgx$ (each nontrivial factor of x is of size at least 2), it can be seen that the worst case running time of R1 is bounded by a polynomial in $(lgx + lgy)$. Note that there is only one call to FACTORIZE which is counted as unit time.

2. Reducing FACTORIZE to FACTOR

```
R2(x) {
            If (x=1) return 1
            Set S = empty set
            While (x >1)
                Use FACTOR and binary search to find a nontrivial factor p of x.
                Set S = S ∪ {p}
                Set x = x/p
            Return S
}
```

The input size is (lgx). It can be seen that the number of calls to FACTOR is bounded by $(lgx)^2$. The while loop will run at most lgx times. In each iteration of the while loop, binary search will call FACTOR at most lgx times. Each call to FACTOR is counted as unit time. Hence the worst case running time of R2 is bounded by a polynomial in (lgx).

D2. [Prove that P ≠ EXP via a "Diagonalization" argument]

Take it granted that there exists an (infinite) listing of all polynomial time algorithms. Let P_k be the k-th algorithm in this listing. Take it granted that there exists a universal program U such that $U(x ; k) = P_k(x)$, for every sting x and number natural number k. The running time of U on input (x ; k) is polynomial in |x| + k. Consider the following algorithm :

```
diag-p (k)

{  If (U(k ;  k) == true)
           Return false
   Else
           Return true
}
```

Show that diag-p is an exponential time algorithm and the language (a.k.a decision problem) it decides is different from any language in P. From this conclude that P ≠ EXP.

Answer. Please recall (the convention specified in lecture) that all inputs to all algorithms are given as binary encodings (unless otherwise specified). Let A_k be a language in P which is decided by P_k. Let D be the language decided by diag-p. Let <k> represent the binary encoding of the natural number k. We can observe from the description of diag-p that,

$$<k> \in A_k \iff (U(k\ ;\ k) == true) \iff <k> \notin D$$

Hence $D \neq A_k$.

We note that the running time of diag-p on input k (which is given as log k bit string) is a polynomial in k, since the running time of U on input (k,k) (which is given as O(log k) bit string) is polynomial in k (and not |<k>| which is log k). Hence diag-p runs in exponential time.

D3. [Understanding P and NP and the input length]

0/1 KNAPSACK problem:

We are given a knapsack with maximum capacity W and a set S consisting of n items. We are also given a number k. Each item i in S has some weight w_i ($\leq$ W) and benefit value b_i =1 (w_i, b_i , W, k are natural numbers which are given in binary encoding).

0/1 KNAPSACK: Decide if the knapsack can be packed with items so that the total benefit value is at least k?

It can be shown that 0/1 KNAPSACK problem is NP-complete.

Professor Smart claimed that he can solve the 0/1 KNAPSACK problem in time $O(nW + lgk)$ (we will also see subsequently in the course a 'dynamic programming' algorithm with same running time). Thus Professor Smart claimed that 0/1 KNAPSACK problem is in P and that he has shown P=NP. Could you find a flaw in his argument?

Answer. Please recall that an algorithm is called polynomial time algorithm if its worst case running time on input x (where x is a binary string) is upper bounded by p(|x|), where p is a fixed polynomial. Please also recall that inputs to algorithms are provided in binary encoding (unless otherwise specified).

Please note that the total input size in binary encoding is $O(n(lgW) + lgk)$. Hence the running time of Professor Smart's algorithm, which is $(nW + lgk)$, is not polynomial in the input size and Professor Smart's algorithm is not a polynomial time algorithm.

D4. [A language and its complement]

For a language L, let $\overline{L}$ be its complement language, that is

$\overline{L}$ = {x : x is a binary string and x $\notin$ L} .

Show that L $\leq_P \overline{L}$ if and only if $\overline{L} \leq_P$ L. Here $\leq_P$ represents Karp reduction.

Answer.

(if) Let us assume that L $\leq_P \overline{L}$. We want to show that $\overline{L} \leq_P$ L. Let R be a polynomial time Karp reduction algorithm from L to $\overline{L}$. Then we know:

$$
\begin{aligned}
& x \in L \iff R(x) \in \overline{L} \\
\Rightarrow\ & x \notin L \iff R(x) \notin \overline{L} \\
\Rightarrow\ & x \in \overline{L} \iff R(x) \in L
\end{aligned}
$$

Therefore R also serves as a polynomial time Karp reduction algorithm from $\overline{L}$ to L. Hence $\overline{L} \leq_P$ L .

(only if) Can be argued similarly as above.