Introduction To Algorithms
CS430

Spring 2014
HomeWork 10
Sample Problem–No need to turn in

1. Problem 34.1-1 **Solution** If LONGESTPATHLENGTH could be solved in polynomial time, then we could solve the decision problem LONGESTPATH in polynomial time for instance $< G, u, v, k >$ by running LONGESTPATHLENGTH on $< G, u, v >$ and returning true if and only if LONGESTPATHLENGTH returned a value $\geq k$. If $LONGESTPATH \in P$, then we identify the length of the longest path by essetially doing a sequential search (binary search would be faster, though still in P) over all possible path lengths.

2. 34.1-3 **Solution** Clearly, an ASCII text file can be represented as a string of bits; that is how one could save some bits by writing the numbers in binary. At any rate, the number of words of memory required for the adjacency-list representation wiil be $O(|V| + |E|)$. Each vertex number will be between 1 and $|V|$, and so will require $\Theta(lg|V|)$ bits to store. So, the total number of bits needed will be$O((|V| + |E|)lg|V|)$ To show that these representations are polynomially related, one can how that one representation can be transform into other representation in a polynomial-time.

   To Convert the matrix to the adjacency list:

   ```
   Repeat for every row
       Scan through the row from left-to-right.
       Add the column number to the list
   ```

   To Convert the adjacency list to matrix:

   ```
   Start with a matrix M of all 0's.
   Let i be a counter of what line you are at. In that line:
       if hit the number j
           then put a 1 at M[i, j]
   ```

3. 34.1-6 **Solution**

   (a) Since L_1 and L_2 are both in P , there exists machines M_1 and M_2 which decides L_1 and L_2, respectively. Design a machine M_3 which takes w as input as the following:
   $M_3(w)$:
   1 Run M_1 with input w. If M_1 accepts w, then accept.
   2 Run M_2 with input w. If M_2 accepts w, then accept.
   It is clear that M_3 accepts w if and only if M_1 or M_2 accepts w and M_3 runs in polynomial time.

(b) Since L_1 and L_2 are both in P , there exists machines M_1 and M_2 which decides L_1 and L_2, respectively. Design a machine M_3 which takes w as input as the following:
$M_3(w)$:
1 Run M_1 with input w. If M_1 accepts w, then run M_2 on w, else reject. 2 If M_2 also accepts w, then accept, else reject. It is clear that M_3 accepts w if and only if both M_1 or M_2 accepts w and M_3 runs in polynomial time.

(c) Since L_1 and L_2 are both in P , there exists machines M_1 and M_2 which decides L_1 and L_2, respectively. Design a machine M_3 which takes $w = a_1a_2a_n$ with each $a_i \in \sigma$ as input, as the following: $M_3(w)$: 1 For $i = 0, 1, 2, ..., n$
2 Run M_1 with input $w_1 = a_1a_2...a_i$ and run M_1 with input $w_2 = a_{i+1}a_{i+2}...a_n$. If both M_1 and M_2 accepts w, then accept.
3 If none of the iterations in Stage 2 accept, then reject.
It is clear that M_3 accepts w if and only if $w = w_1w_2$, where M_1 accepts w_1 and M_2 accepts w_2 . Also, M_3 runs in polynomial time.

(d) Since L_1 is in P , there exists machines M_1 that decides L_1 . Design a machine M_2 which takes w as input as the following:
$M_2(w)$:
1 Run M_1 with input w. If M_1 accepts w, then reject, else accept.
It is clear that M_2 accepts w if and only if M_1 rejects w, and M_2 runs in polynomial time.

(e) Since L_1 is in P , there exists machines M_1 that decides L_1 . By using dynamic programming, a machine M_2 that decides L_1 in polynomial time, which takes $w = w_1w_2...w_n$ as input, can be constructed as follows:
$M_2(w)$: 1 if $w = \epsilon$, then accept
2 for $i = 1$ to n
3 for $j = 1$ to n
4 $A[i, j] = 0$
5 for $i = 1$ to n
6 set $A[i, j] = 1$ if $w_i \in L_1$
7 for $l = 2$ to n
8 for $i = 1$ to $nl + 1$
9 $j = i + l1$
10 set $A[i, j] = 1$ if $w_i...w_jL_1$
11 for $k = i$ to $j1$
12 set $A[i, j] = 1$ if $A[i, k] = 1$ and $A[k, j] = 1$
13 if $A[1, n] = 1$ then accept
14 else reject

Since each step takes polynomial time and there are $O(n^3)$ steps in it, it takes polynomial time for M 2 to decide L_1.

4. Show that $\leq_P$ relation is a transitive relation on problems. That is, show if $P_1 \leq_P P_2$ and $P_2 \leq_P P_3$, then $P_1 \leq_P P_3$. **Solution** Let $P_1 \leq_P P_2$ and $P_1 \leq_P P_2$, i.e.

there exist polynomial-time computable reduction functions $f_1 : \{0,1\}^*\{0,1\}^*$ and $f_2 : \{0,1\}^*\{0,1\}^*$ such that
$x \in P_1 \Leftrightarrow f_1(x) \in P_2$
$x \in P_2 \Leftrightarrow f_2(x) \in p_3$
Define $f3 = f_1 \circ f2$, then L 3 is a polynomial-time computable function : $\{0,1\}^*\{0,1\}^*$
,
and
$x \in P_1 \Leftrightarrow f_3(x) \in P_3$

holds. Hence $P_1 \leq_P P_3$.