CS341 Assignment 4 Marking Scheme

April 11, 2011

1 Question 1

a)

Suppose the problem is decidable. Then there exists a program $SometimesHalts(.)$ that, given a program P as input, decides whether there exists an input w such that $P(w)$ halts. Using $SometimesHalts(.)$, we will construct a program that solves the halting problem. For an instance (P, w) of the halting problem, we can construct the following program:

Reduce(x):
run P on w

The program $Reduce(.)$ always runs P on w regardless of the input it receives. If P halts on w, $Reduce(x)$ halts on all inputs x, otherwise it never halts on any input. We can construct a program for solving the halting problem as follows:

halts(P,w):
construct Reduce(.) as above
return SometimesHalts(Reduce(.))

but the halting problem is undecidable - contradiction.

b)

Analogously to a), suppose there exists a program $gotA(P)$ that decides if P outputs "You got an A..." on input "What is...". We can construct a program $Reduce(.)$ as follows:

Reduce(x):
run P on w, suppressing the output
output "You got an A..."

The above program will output "You got an A..." if and only if P halts on w. Just like in the previous case, running $gotA(Reduce(.))$ decides whether P halts on w - contradiction with the halting problem being undecidable.

c)

Similarly to a) and b), suppose there exists a program $isFinite(P)$ that decides if P accepts a finite set of inputs. Consider the following program:

Reduce(x):
run P on w
accept x

If P halts on w, $Reduce(.)$ accepts all possible inputs (an infinite number). If P does not halt on w, $Reduce(.)$ will never terminate and therefore will not accept any input (it will accept 0 inputs, which is finite). Therefore we can call $isFinite(Reduce(.))$ to solve the halting problem.

Marking scheme: 5 marks for each question. 2 marks off for confusing inputs in b), or other similar minor errors.

2 Question 2

a) This problem is in NP. Given a sequence of vertices in the graph, we can verify that it forms a simple path in $O(k)$ time where k is the length of the path.

To prove the problem NP-complete, we will reduce the problem $HAMILTONIAN-CIRCUIT$ to this problem. That is, given an instance $G = (V, E)$ of $HAMILTONIAN-CIRCUIT$, we will produce an instance (G', k, u, v) of the longest path problem such that the answer to (G', k, u, v) is "yes" iff the answer to G is "yes".

Let w be an arbitrary vertex of G. Create G' as a copy of G with an extra vertex w' which is adjacent to every neighbour of w. Let k be the number of vertices in G and let $u = w$ and $v = w'$. If there is a hamiltonian circuit in G, then there exists a path of length k between w and w' which can be found by traversing the hamiltonian circuit starting from w. Conversely, if there exists a path of length k between w and w' then by identifying w and w' we obtain a hamiltonian circuit in G. Thus solving the longest path problem for an instance generated this way gives us the answer to the corresponding instance of $HAMILTONIAN - CIRCUIT$.

The above transformation can be done in polynomial time, which concludes the proof. If we had a polynomial-time algorithm for the longest path problem, we could solve (in polynomial time) the hamiltonian circuit problem, and since we know that the hamiltonian circuit problem is NP-complete, we can conclude that we would be able to solve all the problems in NP.

b)

This problem is in NP. Given a set of k vertices in the graph, we can verify in $O(k^2)$ time that there is no edge between any two vertices in the set.

To prove that it is NP-complete, we will reduce $CLIQUE$ to this problem.

Given an instance (G, k) of $CLIQUE$, we produce an instance (G', k) of $INDEPENDENT-SET$ where G' is the complement of the graph G. That is, $V(G) = V(G')$ and for each pair of vertices u, v, we have $\{u, v\} \in E(G')$ iff $\{u, v\} \notin E(G')$. It is easy to see that each clique in G is an independent set in G' and, conversely, each independent set in G' is a clique in G.

Taking the complement of the graph can be carried out in $O(|V|^2)$ time, which is clearly polynomial.

Marking scheme: 10 marks for every question. 6 marks off for choosing a single, arbitrary pair of connected vertices as u,v in a). 8 marks off in a) and 4 in b) for reducing in the wrong direction. 1 mark off for not showing membership in NP.

Note that there were many possible correct solutions to this question, some very different from the above. In particular, a) could be solved by reducing $HAMILTONIAN - PATH$ to the longest path problem. b) could be solved by reducing $VERTEX - COVER$ or $3SAT$ to the independent set problem.

3 Question 3

a) This problem is in P. First, take the subgraph of G induced by $V \backslash L$ and construct its spanning tree T' using Kruskal's algorithm. If no such tree exists, return false. For each vertex in L, check if it has a neighbour in $V \backslash L$. If it does not, return false. If all vertices in L have neighbours in $V \backslash L$, return true.

b) This problem is NP. Given a set of edges, we can verify in polynomial time if this set forms a spanning tree for G and if L is the set of its leaves.

To prove it is NP-complete, we will reduce $HAMILTONIAN - CIRCUIT$ to this problem. Given an instance G of $HAMILTONIAN - CIRCUIT$, create G' as a copy of G with one extra vertex w' whose neighbours are the same as the neighbours of some vertex w in G (same as in Question 2a). Let $L = \{w, w'\}$. It is easy to show that a tree with 2 leaves must be a path, and that a spanning tree with 2 leaves must be a hamiltonian path for G'. Such a path exists in G' if and only if a hamiltonian circuit exists in G, by arguments similar to those in the proof of 2a).

c) This problem is NP-complete. For $|L| = 2$, this problem is identical to problem 3b), so the proof proceeds analogously.

Marking scheme: 5 marks for each question. 2 marks for the correct complexity class, 3 marks for the proof.

4 Office Hours

The office hours for complaints and questions about the final will be held on Monday 11 April from 2 to 4 pm, and on Tuesday 12 April, from 11 to 2 pm, in DC2501.