Answers to 527 final, Spring 2004

Burton Rosenberg

May 17, 2004

5.7.1: Show L R.E. iff exists NTM M, $L = \{ w : w \text{ output on empty tape} \}$.

If L is R.E. there is a TM M' which Turing enumerates L. The NTM M non-deterministically decides whether to halt when M' enters the special enumerator state. All and only $w \in L$ can appear as the result of M.

In the other direction, given M as described we have to "rephrase" the computation according to some accepted definition of R.E.. Let M' enumerate all hints to M, and simulate M on each hint. At the end of the hint, if M is halted then M' enters the special state, else it does not. In any case it continues to the next hint. Therefore M' Turing enumerates L.

5.7.3.: Assume $L \subseteq \Sigma^*; \Sigma^*$ is R.E. Let $L' = \{ x \in \Sigma^* : x; y \in L, \text{for some } y \in \Sigma^* \}$. Show L' is R.E.

Let M Turing enumerate L. Construct M' to Turing enumerate L' as follows. M' lets M run. When M enters the special state with $x; y$ on the tape, M' copies x to its tape and enters its special state.

(b) No, L can be recursive while L' R.E. For instance, $x; y$ could be that y is an accepting computation of a TM M' on input x. This is recursive (you can verify a computation correct without "magic"). However L' would be the language of M', which could be R.E.

6.1.1.: Show P-time is closed by union, intersection and concatenation.

Let M_i be a P-time TM accepting L_i, for $i = 1, 2$.

To accept $L = L_1 \cup L_2$, try $M_1(x)$ and $M_2(x)$ and accept if either does. The time is the sum of the individual times, hence polynomial.

For intersection, accept only if both do. Again, the time is the sum of the individual times, hence polynomial.

For concatentation, try all possible decompositions of x as x_1 and x_2, accepting if $M_1(x_1)$ and $M_2(x_2)$ both accept. The time is $|x|(f_1(|x|) + f_2(|x|)$, hence polynomial.

6.2.1.: A graph is Eulerian iff it is connected and every vertex has out-degree equal to in-degree.

It is clear that if the connected and degree requirements are not met, there can be no tour.

Suppose the conditions are met. Take any vertex and begin to travel. Since out-degree equals in-degree you can always take another step. Eventually you will repeat a vertex for the first time. Consider the circuit C created when this path first closes. Remove the edges on C from the graph. By induction get a Eulerian tour E of the remaining edges. Since the graph is connected, if there are any edges remaining after the removal of the edges of C then some vertex on C must have in-degree greater than one — hence E and C intersect. Surgery the two paths together.

6.2.3.: Find a solution the the Traveling Salesman problem on the given 5 city dataset.

A, C, B, E, D for 13. The tour goes from D to A and is 18.

6.2.5. Given the graph, you show a Hamiltonian cycle, a max clique (of 3), an Independent set (of size 10), and a node cover (of size 12). For full credit you should show them.

6.3.1.: Show all satisfying assignments for,

$$(x_1 \vee \neg x_2 \vee x_3), (\neg x_1 \vee x_4), (x_2 \vee \neg x_3 \vee \neg x_4)$$

There are eight:

x_1	x_2	x_3	x_4
T	T	T	T
T	T	F	T
T	F	F	T
F	F	F	F
F	F	T	F
F	F	F	T
F	T	T	T
F	T	T	F

6.3.3.: Proof of 2-SAT in P-time using reachability in graphs.

See separate write-up.

6.4.1.: Show NP is closed by union, intersection, concatentaion and Kleene star.

Let L_i be recognized by NTM M_i in P-time, $i = 1, 2$.

The machine recognizing $L_1 \cup L_2$ takes two hints and runs M_1 and M_2 on the input and the hints, accepting if either does.

The machine recognizing $L_1 \cap L_2$ akes two hints and runs M_1 and M_2 on the input and the hints, accepting if both do.

The machine recognizing $L_1 \circ L_2$ takes three hints y_1, y_2, y_3, uses the first to parse the input $x = x_1 x_2$, and the second two to run $M_1(x_1, y_2)$ and $M_2(x_2, y_3)$, accepting if both do.

The machine recognizing L_1^* takes a various number of hints $y_1, y_2, \ldots, y_{k+1}$, uses the first to parse the input $x = x_1 x_2 \ldots x_k$, and the rest to verify membership $M_1(x_i, y_{i+1})$ for $i = 1, \ldots, k$, accepting if all sub-computations do.

The time complexity of all these constructions is bound by the sum of the complexities of the M_i plus some overhead linear in $|x|$, except for Kleene star which requires at most $|x|$ repetitions of M_1. In all cases, this is polynomial in $|x|$.

7.1.1.: Show 3-COLOR is in NP and give a P-time reduction to SAT.

To show 3-COLOR is in NP we can be given a coloring and check if it is correct. This requires looking at each edge and checking that the endpoints are different colors. This is linear in the input size (number of edges in the graph).

Reduction to SAT: we need to write down a SAT equation which is satisfied if and only if the given graph is 3-colorable. For every node v_i in the graph, we have three variables v_{i1}, v_{i2}, v_{i3}. Exactly one is true and represents the color of the node. These clauses assure exactly one is true:

$$(v_{i1} \vee v_{i2} \vee v_{i3}), (\neg v_{i1} \vee \neg v_{i2}), (\neg v_{i2} \vee \neg v_{i3}), (\neg v_{i1} \vee \neg v_{i3})$$

for all i such that v_i is a vertex. For each vertex this generates a constant times the size needed to represent a variable and can be done in a linear time scan of the input. For every edge we check that the endpoints have different colors.

$$(\neg v_{i1} \vee \neg v_{j1}), (\neg v_{i2} \vee \neg v_{j2}), (\neg v_{i3} \vee \neg v_{j3}),$$

for all i, j such that (v_i, v_j) is an edge in the graph. For every edge this generates a constant times the size needed to represent a variable and can be done in a linear time scan of the input.

Suppose there is a 3-coloring. Setting v_{ik} true if node i is color k, and false otherwise, we have satisfied the clauses. Conversely, if the clauses are satisfied, for all i exactly one k has v_{ik} true, and it can be the color of node i.

7.3.1.: Show that EXACT-COVER remains NP complete when restricted so that no covering set has more than 3 elements, and that no element is in more than 3 covering sets.

We transform a general instance so that it conforms to the restricted version. We first transform the covering sets. Let a cover $c = \{ u_1, \ldots, u_k \}$ where $k > 3$. We create a whole bunch of subsets which chain together small subsets so that they all must be taken or none take, reproducing the effect of c. Introduce elements x_i, y_i, z_i for $i = 1, \ldots, k$. The new sets replacing c are.

$$\{y_i, u_i\}, \{z_i, x_{i+1}\}, \{x_1\}, \{z_k\}$$

and also

$$\{x_i, y_i, z_i\}$$

The point of this is that either c is taken or not in the original. If it is than all of the u_i are covered, else none are covered. It is possible to take all the sets $\{y_i, u_i\}$ and cover the remaining x_i and y_i with the other sets; it is also possible to take none of the $\{y_i, u_i\}$ and cover all the x_i, y_i, z_i with the sets $\{x_i, y_i, z_i\}$. It not possible however to take some of the $\{y_i, u_i\}$ and not others. It is all or nothing.

It is possible to modify this construction. Instead of $\{y_i, u_i\}$ use sets $\{y_i, u_i, u_i'\}$ so that some widgets cover two elements of c. Here is an example,

```
+---+---+         +---+---+         +---+---+
|   |   |         |   |   |         |   |   |
x1  y1  z1 ----- x2  y2  z2 ----- x3  y3  z3
    |                 |                 |
 +--+--+           +--+--+             u5
 |     |           |     |
 u1    u2          u3    u4
```

Not shown are the singleton sets $\{x_1\}$ and $\{z_3\}$.

Replacing every covering set with this construction we have an equivalent problem but no set has more than three elements. Formally, we have a reduction that for every c writes out a collection of sets. Their is an exact cover to the original problem if and only if there is an exact cover to the rewritten problem.

We next transform a general instance of exact cover so that no element appears in more than 3 covers. To do this we create a family of sets which cover elements so that among the $u_1, \dots, u_k$ elements exactly one is uncovered. Any element u in more than 3 covers is replaced by this construction, replacing a use of u with a use of u_i.

The construction addes elements and covers. The covers have 2 elements each, so you could draw the construction as a graph and a cover as an edge cover. Create elements u_i, v_i, w_i for $i = 1, \dots, k$ and glue elements g_i for $i = 1, \dots, k-1$. The covers are,

1. $\{w_i, v_i\}$ and $\{v_i, u_i\}$ for $i = 1, \dots, k$,
2. $\{w_i, g_i\}$ for $i = 1, \dots, k-1$,
3. and $\{g_{i-1}, w_i\}$ for $i = 2, \dots, k$.

Here is an example,

```
w1 ----- g1 ----- w2 ----- g2 ----- w3 ----- g3 ----- w4
|                 |                 |                 |
v1                v2                v3                v4
|                 |                 |                 |
u1                u2                u3                u4
```

To cover all glue elements g_i k edges must be included from the backbone, leaving exactly one w_i uncovered. This is covered by the edge to the corresponding v_i and u_i is uncovered. The remaining v_j are covered by edges to u_j so these are not available for covering by the cover in which the u_j is used.

(b) We can consider what happens when the restriction is to 2 elements in a cover, or 2 uses of the elements in any cover.

If $|c| \leq 2$ for all covers c, the result covers can be represented as edges in a graph. We are looking for a collection of edges of minimum size such that each vertex is covered by the selected edge set exactly once. This problem is called EDGE COVERING, and is in P.

If there can be at most two c_i such that any u is in these c_i, then the problem can be reduced to 2-SAT and is therefore P-time. The reduction is fairly direct. Each cover is represented by a variable and is true if the cover is taken. Clauses express when two covers cannot be taken simultaneously. If c_i and c_j intersect than $(\neg c_i \vee \neg c_j)$ forces that both variables c_i and c_j cannot be true. The condition for covering is that if $u \in c_1, c_2$ then $(c_1 \vee c_2)$. It is here we use the 2 condition to get 2-SAT.

There is an interested duality in our constructions. To reduce the size of cover sets we used a construction which required sets of size 3 but no element was in more than 2 covers. To reduce the number of covers which a set was in we used a construction that required sets of size 2 but some elements are in 3 sets.

7.3.3.: Show that HAM-PATH is NP complete, and that HAM PATH SPEC VERTICES is NP complete.

We do this with one construction with a reduction from HAM CYCLE. Showing that HAM PATH is in NP is easy.

Given an instance of HAM CYCLE we modify the graph so that the new graph has a HAM PATH if and only if the original had a HAM CYCLE. Create a sink vertex s_0 and a source vertex s_1. Every edge (x, y) is subdivided by two new vertices u and v into $(x, u), (u, v), (v, y)$. Add edges (u, s_0) and (s_1, v). The HAM CYCLE is now a path by replacing exactly one step of the cycle by a route to the sink and from the source. That is, if the original graph used (x, y) in the cycle than the new graph uses $(x, u), (u, s_0)$ as the end of the path and $(s_1, v), (v, y)$ as the start of the path.

Conversely, if there is a HAM CYCLE in the new graph, it must start at s_1 and end at s_0. If s_1 goes to v the only way to cover u is to have u go to s_0. so (x, u) and (v, y) are in the path. These can be replaced by (x, y) in the original to get a cycle.

This construction might triple the size of the graph: every edge introduces two new vertices. But no matter, it is still polynomial.

7.3.5.: Show that IND SET is NP complete even if K is fixed to $\lceil n/2 \rceil$.

We transform a general instance of IND SET so that there are always enough independent vertices "given for free" so that K needn't be small.

Given a graph G and a bound K, add a set of vertices V with no incident edges. In the new graph G' all the V are in the IND SET, so that the problem is the same in G' if $K' = K + |V|$. We set this equal to $\lceil n/2 \rceil$, where $n = |G| + |V|$. So $|V| = |G| - 2K$.