CS140: Algorithms

Spring 2001

Solution to Assignment 8

1. Recall that a *finite automaton* is a directed graph with edges labelled by symbols from an alphabet Σ. In addition, one node of the graph, s, is specified to be the *start state* and one or more nodes are specified as accept states. A word w in Σ^* is *accepted* by the DFA if there is a path with edges labels w from s to some accept state. We use $L(M)$ to denote the language of the automaton M; i.e. set of strings accepted by M.

   A finite automaton is a *deterministic* finite automaton (*DFA* if for every node v in the graph and every symbol $a \in \Sigma$, v has exactly one out-edge labelled a.

   (a) Draw a DFA for the language. $\{0 * 1\}$.
   (Left to the student.)

   (b) Given a DFA M and an integer K, show how to construct a DFA that accepts exactly those string of $L(M)$ that have length K.

   The new machine will have states (q, i), where q is any state in M and i is any integer in the range $[0, K]$. For any transition from p to q on a in M we add transitions from (p, i) to $(q, i+1)$ on a for $i = 0, ..., K-1$. The start state of the new machine in $(s, 0)$, where s is the start state of M. For any accept state p in M, the state (p, K) is an accept state in the new machine.

   (c) Design an algorithm that takes as input the description of a DFA M and an integer K and outputs the number of strings of length K in $L(M)$. (Hint: Use inductive design, strengthening the inductive hypothesis as necessary. Then use dynamic programming to achieve polynomial time.)

   Given M and K we first construct the DFA described in the previous section. Then the strings in $L(M)$ of length K exactly correspond to the paths in the new DFA between the start state and an accepting state. Thus it suffices to count these paths.

   To do so we can assume we counted the number of paths between $(s, 0)$ and every node of the form $(p, K-1)$. Denote this count as $N(s, p, K-1)$. Then we can compute $N(s, q, K) = \sum_{nodesp} N(s, p, K-1) * E(p, q)$, where the sum is taken over all nodes of the form $(p, K-1)$ and $E(p, q)$ is the number of transitions from state $(p, K-1)$ to (q, K).

   We have to compute $N(s, p, i)$ for n states and K values of i. Each value takes $O(n)$ time to compute so the running time is $O(n^2 K)$.

2. Problem 16-4 on page 326 of the text.

   This is a variation of weighted independent set, which was discussed in class.

3. Problem 16-5 on page 327 of the text.

   Let $S(i)$ denote the set of nodes v such that there is a path labeled $\sigma_1, \ldots, \sigma_i$ from v_0 to v. Then $S(0) = \{v\}$ and $S(i+1)$ is the set of nodes that have an in-edge labeled σ_{i+1} from a node in $S(i)$. Then there is a path labeled with the sequence s from v_0 if and only if $S(k)$ is non-empty.

   We can compute $S(0)$ in constant time. Given $S(i)$ we can compute $S(i+1)$ in time $O(m)$, where m is the number of edges in the digraph. Thus the running time is $O(km)$.

   For part (b) we also compute for each $v \in S(i)$ the probability of the most probable path from v_0 to v that is labeled $\sigma_1, \ldots, sigma_i$. The update step is modified accordingly.