Illinois Institute of Technology
Department of Computer Science

Solutions to Third Examination

CS 330 Discrete Structures
Fall Semester, 2013

47 students took the exam; the statistics were:

Minimum	0
Maximum	98
Median	53
Average	55.28
Std Dev	24.57

1. Professor Reingold Goes to Boston

   Professor Reingold is driving to Boston for his grandchild's birthday party. Because he suffers from a weak bladder, he would like to stop at highway exits as often as possible on the way. He realizes, though, that he must travel at least n miles between stops in order to arrive in Boston in time for the party. His map details all highway exits on his route and the distances between them.

   (a) Give a greedy algorithm for finding the exits at which Professor Reingold should stop so as to maximize the number of stops he makes.

   Sort the exits by distance from the starting point. Until Washington is reached, repeatedly stop at the closest exit at least n miles from the current location.

   (b) What is the running time of your algorithm if there are m highway exits along the route?

   The list of exits can be sorted in $\Theta(m \log m)$ time. Once the list is sorted, each exit is considered exactly once (and is either passed or taken), so this stage of the algorithm takes $\Theta(m)$ time. Thus the algorithm runs in time $\Theta(m \log m)$.

   (c) Argue that your algorithm yields an optimal solution.

   We prove this by contradiction. Let $G = \langle e_0, e_1, e_2, \ldots \rangle$ be the sequence of exits chosen by the greedy algorithm. Let e_j be the greedy algorithm's first mistake; that is, there is an optimal solution beginning with $\langle e_0, e_1, e_2, \ldots, e_{j-1} \rangle$, but there is no optimal solution beginning with $\langle e_0, e_1, e_2, \ldots, e_{j-1}, e_j \rangle$. Consider an optimal sequence of exits $O = \langle e_0, e_1, e_2, \ldots, e_{j-1}, e'_j, e'_{j+1}, \ldots \rangle$. Since the greedy algorithm always selects the first exit at least n miles from the current location, clearly e_j occurs earlier than e'_j. It follows that, if e_j were substituted for e'_j in O, e'_{j+1} would remain a valid successor, as if at least n miles separate e'_j from e'_{j+1}, certainly at least n miles separate e_j from e'_{j+1}. Since this new sequence of exits is also optimal, this contradicts our claim that e_j was the greedy algorithm's first mistake. Thus the greedy algorithm yields an optimal solution.

2. Sources

   A node $s \in V$ of a directed graph $G = (V, E)$ is called a *source* if it has no incoming edges. Given the adjacency structure of a graph, design *and analyze* an $O(|V| + |E|)$ algorithm to determine whether it has a source or not.

   Systematic examination of the vertices in the adjacency structure achieves the $O(|V| + |E|)$ time bound: Add a mark bit to each vertex in the graph. Then, just go down the list of vertices in the adjacency structure, for each edge in the edge list, mark the vertex to which the edge goes; at the end of the process, if any vertex is unmarked, it is a source.

3. BFS/Finite State Machines

   Consider a regular language L over the 26 letters of the English alphabet, and a finite state machine M accepting it. Assume that M has at least 3 states and that its transition diagram has no self-loops or parallel edges.

   (a) Describe *and analyze* an algorithm for finding a shortest string in L.

   Let S be the set of states. Consider the transition diagram of M as a graph in which the vertices are the states and the transitions are the edges, run breadth-first search from the start state s_0 until a state in $f \in F$ is found, yielding a shortest path from s_0 to f, and abort the search there. Using the predecessor values to trace the shortest path backward, return the string formed by the labels of the transitions along the shortest path from s_0 to f.

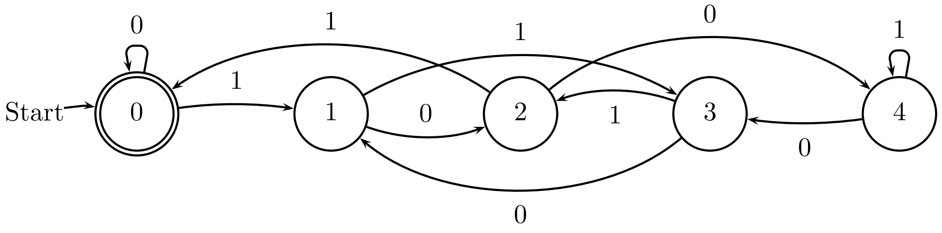
   Since BFS runs in $O(|V| + |E|)$ time on a graph $G = (V, E)$, and this graph has $|S|$ vertices and $|\Sigma| \times |S| = 26|S|$ edges, the algorithm runs in $O(|S|)$ time.

   (b) Prove that the transition diagram of M is not planar.

   By Euler's formula a planar graph has $|E| \leq 3|V| - 6$. As before, $|V| = |S|$ and $|E| = 26|S|$. Thus, if the graph were planar, we would have $26|S| \leq 3|S| - 6$, which could only hold if $|S| \leq -6/23$. Clearly, though, $|S| \geq 0$ (in fact, we are told that $|S| \geq 3$), leading to a contradiction. Thus the graph must be non-planar.

4. Finite State Machines

   In the lecture of November 18, the following machine was presented to recognize multiples of 5 written in binary and read from left to right (that is, most significant bit first):



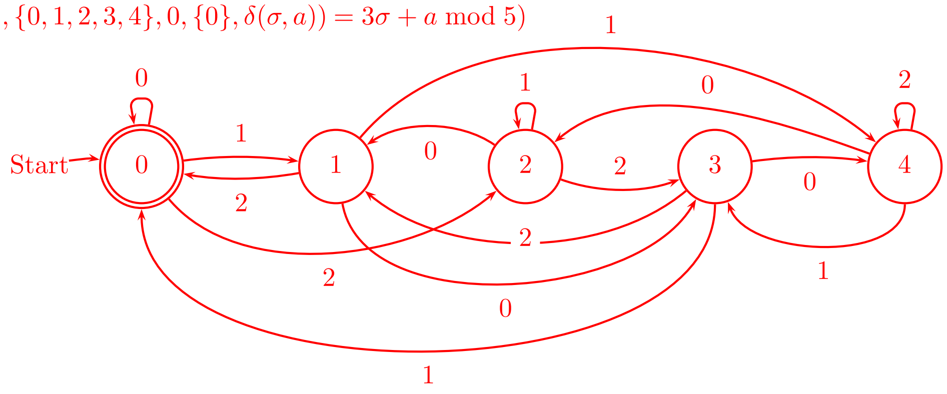
   (a) Give the formal specification (quintuple) for this FSM.

   $(\{0, 1\}, \{0, 1, 2, 3, 4\}, 0, \{0\}, \delta(\sigma, a)) = 2\sigma + a \bmod 5)$

   (b) Draw and give the formal specification of the equivalent FSM which recognizes the same set of numbers (that is, multiples of 5) written in ternary (base 3).

   The changes are to the input alphabet and the transition function:

   $(\{0, 1, 2\}, \{0, 1, 2, 3, 4\}, 0, \{0\}, \delta(\sigma, a)) = 3\sigma + a \bmod 5)$

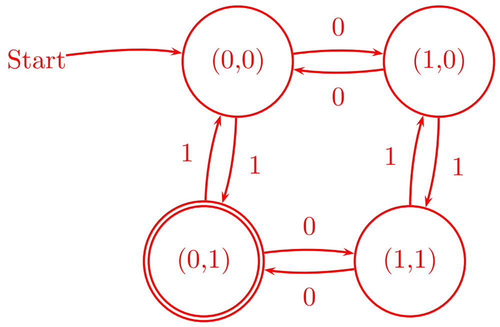


5. Regular Languages

   Indicate whether the following sets are regular or not, and justify your answer.

   (a) Strings with an even number of zeroes and an odd number of ones.

   This is essentially the same as exercise 36 on page 876 of Rosen, assigned in HW 7. We give a 4-state FSM that keeps track separately of the parity of the numbers of zeros and ones. $(\{0,1\}, \{0,1\} \times \{0,1\}, (0,0), \{(0,1)\}, \delta(\sigma,(a,b)) = (\sigma + a + 1 \bmod 2, \sigma + b \bmod 2))$. δ just flips the first component on a 0 and flips the second component on a 1. Pictorially,



   (b) Strings *not* in the Oxford English Dictionary.

   The number of strings in the dictionary is large but finite. We know that any finite set is regular. We also know that regularity is closed under complementation. Thus the set of strings not found in the OED is regular.

   (c) $L = \{a^n b^m | n \neq m\}$. (*Hint*: If L is regular, is $(\Sigma^* - L) \cap a^*b^*$ also regular?)

   The proof is by contradiction. Suppose L were regular. Following the hint, we note that because regularity is preserved by complementation, L is regular if and only if $\Sigma^* - L$ is regular. Because a^*b^* is a regular expression, the set of strings that it describes is regular. Finally, the intersection of two regular sets is regular, so L is regular if and only if $(\Sigma^* - L) \cap a^*b^*$ is regular. But $(\Sigma^* - L) \cap a^*b^* = \{a^n b^m | n = m\}$ which is not regular, as we saw in class on both November 18 and November 20 (by the pigeonhole principle).