CS515: Algorithms and Data Structures, Fall 2014

Homework 2*

Due: Mon, Oct/27/14

Homework Policy: Students should work on homework assignments in group of preferably three people. Each group submits to TEACH one set of typeset solutions, and hands in a printed hard copy in the class. You are allowed to discuss the homework with other groups, however, you must mention their names in your submission. Also, you must cite any other source that you use.

The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.

Problem 1. Recall that a deterministic finite automaton (DFA) is formally defined as a tuple $M = (\Sigma, Q, q_0, F, \delta)$, where the finite set Σ is the input alphabet, the finite set Q is the set of states, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of final (accepting) states, and $\delta : Q \times \Sigma \to Q$ is the transition function. Equivalently, a DFA is a directed (multi-)graph with labeled edges, such that each symbol in Σ is the label of exactly one edge leaving any vertex. There is a special *start* vertex q_0, and a subset of the vertices are marked as *accepting states*. Any string in Σ^* describes a unique walk starting at q_0; a string in Σ^* is accepted by M if this walk ends at a vertex in F. Stephen Kleene proved that the language accepted by any DFA is identical to the language described by some regular expression. This problem asks you to develop a variant of the Floyd-Warshall all-pairs shortest path algorithm that computes a regular expression that is equivalent to the language accepted by a given DFA. Suppose the input DFA M has n states, numbered from 1 to n, where (without loss of generality) the start state is state 1. Let $L(i, j, r)$ denote the set of all words that describe walks in M from state i to state j, where every intermediate state lies in the subset $\{1, 2, \ldots, r\}$; thus, the language accepted by the DFA is exactly

$$\bigcup_{q \in F} L(1, q, n)$$

Let $R(i, j, r)$ be a regular expression that describes the language $L(i, j, r)$.

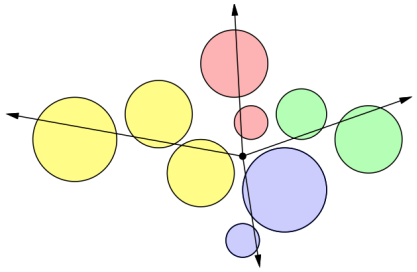
(a) **[5 pts]** What is the regular expression $R(i, j, 0)$?

(b) **[15 pts]** Write a recurrence for the regular expression $R(i, j, r)$ in terms of regular expressions of the form $R(i', j', r - 1)$.

*Problems are from Jeff Erickson's lecture notes. Looking into similar problems from his lecture notes on dynamic programming, all pairs shortest paths and greedy algorithms is recommended.

(c) **[20 pts]** Describe a polynomial-time algorithm to compute $R(i, j, n)$ for all states i and j. (Assume that you can concatenate two regular expressions in $O(1)$ time.)

Problem 2. Given a set C of n circles in the plane, each specified by its radius and the (x, y) coordinates of its center, compute the minimum number of rays from the origin that intersect every circle in C. Your goal is to find an efficient algorithm for this problem.



(a) **[20 pts]** Suppose it is possible to shoot a ray that does not intersect any balloons. Describe and analyze a greedy algorithm that solves the problem in this special case.

(b) **[10 pts]** Describe and analyze a greedy algorithm whose output is within 1 of optimal. That is, if m is the minimum number of rays required to hit every balloon, then your greedy algorithm must output either m or $m + 1$.

Problem 3. Consider the following algorithm for finding the smallest element in an unsorted array:

```
RandomMin A[1..n]
  min ← ∞
  for i ← 1 to n in random order do
    if A[i] < min then
      min ← A[i]          (*)
    end if
  end for
  return min
```

(a) **[5 pts]** In the worst case, how many times does RandomMin execute line $(*)$?

(b) **[10 pts]** What is the probability that line $(*)$ is executed during the nth iteration of the for loop?

(c) **[15 pts]** What is the *exact* expected number of executions of line $(*)$?