Algorithms Fall 2016 Homework 10 Solution

November 29, 2016

PROBLEM 1

Since all the characters in the pattern P are different, when we observe a partial match, there will be no other match overlapping with it. We can just start looking for the pattern from the end of the partial match. The pseudo-code is as follows:

```
ACCELERATED-NAIVE-STRING-MATCHER(T, P)
n = T.length
m = P.length
s = 0
while s ≤ n − m
        for t = 1 to m
                if P[t] ≠ T[s + t]
                        s = s + t
                        break
                else  if t == m
                        print "Pattern occurs with shift" s
                        s = s + t
```

PROBLEM 2

We can move the search pattern from the topmost left corner of the $n \times n$ array, one column at a time, row by row, until the pattern has been compared all $m \times m$ submatrix in T. To extend the original Rabin-Karp algorithm to this problem, we need to make following changes:

- We need to calculate the corresponding value for a matrix of characters instead of a string.
- We need to update the corresponding value when moving the matrix by column.
- We need to update the corresponding value when moving the matrix by row.

The pseudo-code is as follows:

```
2D-RABIN-KARP-MATCHER(T, P, d, q)
p = HASH(P, d, q)
tR = HASH(T, d, q)
for i = 0 to n − m
        t = tR
        for j = 0 to n − m
                if t == p
                        if P[1...m, 1...m] == T[i + 1...i + m, j + 1...j + m]
                                print "Occurrence of pattern is found"
                t = SHIFT − RIGHT(t, T, d, q, i, j, m)
        tR = SHIFT − DOWN(tR, d, q, T, i, m)
```

```
HASH(S, d, q)
hash = 0
for i = 1 to m
        rowHash = 0
        for j = 1 to m
                rowHash = (d * rowHash + S[i][j])  mod q
        hash = (d * hash + rowHash)  mod q
return hash
```

```
SHIFT-RIGHT(hash, T, d, q, row, col, m)
leftColHash = 0
rightColHash = 0
for i = 1 to m
        leftColHash = (leftColHash + T[row + 1][col] * d^(2m-1-i))  mod q
        rightColHash = (rightColHash * d + T[row + i][col + m])  mod q
hash = ((hash - leftColHash) * d + rightColHash)  mod q
return hash
```

```
SHIFT-RIGHT(hash, T, d, q, row, m)
topRowHash = 0
bottomRowHash = 0
for i = 1 to m
        topRowHash = (topRowHash + T[row][i])  mod q
        bottomRowHash = (d * bottomRowHash + T[row + m][i])  mod q
hash = ((hash - topRowHash * d^(m-1)) * d + bottomRowHash)  mod q
return hash
```

PROBLEM 3

We define the string-matching automaton that corresponds to a nonoverlappable pattern $P[1...m]$ as follows:

- The state set Q is $\{0, 1, ..., m\}$
- The start state q_0 is state 0.
- State m is the only accepting state.
- The transition function is defined by the following equation, for any state q and character a:

$$\delta(q, a) = \begin{cases} q+1 & \text{if } q < m \text{ and } p_{q+1} = a \\ 1 & \text{if } p_1 = a \\ 0 & \text{otherwise} \end{cases}$$

There are three cases in the transition function. Since the first case is obvious, we explain the latter two cases. Let $k = \delta(q, a) = \sigma(P_q a)$. We firstly prove that k must be less or equal to 1 by contradiction: suppose $k > 1$, then P_{k-1} must be a suffix of P_q, this contradicts to that P is nonoverlappable. If $p_1 = a$, since P is nonoverlappable, P_1 is the longest prefix of P which is a suffix of $P_q a$, therefore $k = 1$, otherwise $k = 0$.

PROBLEM 4

We can divide P into sub patterns $P_1, P_2, ..., P_K$ by gap character, then build corresponding automatons $M_1, M_2, ..., M_k$ for these sub patterns. Then we connect these automatons to build automaton M as follows:

- For adjacent automatons M_i and M_{i+1} where $1 \le i < k$, we combine the accepting state $q_0^{(i)}$ of M_i with the starting state $m^{(i+1)}$ of M_{i+1} into one state $t^{(i)}$. The state set of M' is $Q_{M'} = (Q_{M_1} - \{q_0^{(1)}\}) \cup (Q_{M_2} - \{q_0^{(2)}, m^{(2)}\}) \cup ... \cup (Q_{M_{k-1}} - \{q_0^{(k-1)}, m^{(k-1)}\}) \cup (Q_{M_k} - \{q_0^{(k)}\}) \cup \{t^{(1),...t^{(k-1)}}\}$.

- Select the start state of M_1 as the start state of M'.
- Select the accept state of M_k as the accept state of M'.
- $\Sigma_{M'} = \Sigma_{M_1} \cup ... \cup \Sigma_{M_2}$
- Add following transitions to the transition function:
  - For all the transitions to an accepting state $m^{(i)} (1 \le i < k)$, we map them to the corresponding merged state $t^{(i)}$.
  - For all the transitions from a starting state $q_0^{(i)} (1 < i \le k)$, we map them to start from the corresponding merged state $t^{(i-1)}$.
  - For any state q in M_i, if character $a \notin \Sigma_{M_i}$, $\delta(q, a) = t^{(i-1)}$.