CSC236 Assignment 3 Solutions

(please note that while we provide a unique solution, others are possible though not all are correct!)

Exercise 1. Proof of correctness for iterative algorithms.

(a) Design an iterative closest pair algorithm for finding the closest pair of points in 2D.

Precondition: Input is a list of n points in the form (x_i, y_i), where $x_i, y_i \in \mathbb{R}$, and $n \geq 2$.

Postcondition: Return a closest pair of points.

Answer. Note that for any two nonnegative real numbers $0 \leq r, s \in \mathbb{R}$ we have

$$\sqrt{r} < \sqrt{s} \iff r < s.$$

Therefore, to compare distances it is enough to compare the square of distances[1].

Suppose that $p = [p_1, p_2]$ and $q = [q_1, q_2]$ represent two points in 2D (so that $p_1, p_2, q_1, q_2 \in \mathbb{R}$). Define

procedure $\text{SQDIST}(p, q)$ ▷ The square of the distance of p and q
    return $(p[0] - q[0])^2 + (p[1] - q[1])^2$

It is clear that SQDIST runs in constant time. We are now ready for the minimum distance algorithm.

procedure $\text{MINDIST}(\text{L})$ ▷ L is a list of 2D points.
    $min \leftarrow \text{SQDIST}(L[0], L[1])$
    $pair \leftarrow [0, 1]$ ▷ pair with minimum distance so far
    $p \leftarrow 0$ ▷ index of one point
    while $p < len(L) - 1$ **do**
        $q \leftarrow p + 1$ ▷ index of other point
        while $q < len(L)$ **do**
            $d \leftarrow \text{SQDIST}(L[p], L[q])$ ▷ square of the distance between one point and other point
            if $d < min$ **then**
                $min \leftarrow d$
                $pair \leftarrow [p, q]$
            $q \leftarrow q + 1$
        $p \leftarrow p + 1$
    return $[L[pair[0]], L[pair[1]]]$ ▷ pair of points with minimal distance

[1] This is a useful trick which one encounters often

(b) Find the complexity class.

Answer. Suppose that the list of points L is of length n. If $n < 2$, the it is obvious the algorithm runs in constant time. Otherwise,

- The initialization in lines 2-7, takes constant time c_1.
- The outer loop, lines 8-17, runs $n-1$ times. On its j-th run (for $1 \leq j \leq n-1$), it performs constant time operations on lines 8, 9, 16, which we denote by c_2; and invokes the inner loop on lines 10-15.
- The inner loop runs $n-j$ times (from $j+1$ to n). In each of its runs it performs only constant-time operations, which we represent as c_3.

The total running time $T(n)$ is therefore:

$$T(n) = c_1 + \sum_{j=1}^{n-1} \left(c_2 + \sum_{k=j+1}^{n} c_3 \right) = c_1 + c_2(n-1) + c_3 \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} 1$$

$$= c_1 + c_2(n-1) + c_3 \left(\sum_{2}^{n} 1 + \sum_{3}^{n} 1 + \cdots + \sum_{n}^{n} 1 \right)$$

$$= c_1 + c_2(n-1) + c_3((n-1) + (n-2) + \cdots + 1)$$

$$= c_1 + c_2(n-1) + c_3 \frac{n(n-1)}{2}.$$

We see that the total running time is $\Theta(n^2)$.

(c) Prove correctness.

Answer. Consider the inner and outer loop invariants:

Lemma 1.. *Suppose and the outer loop executes at least i times. Then $P(i)$: at the end of the i-th iteration, $p = i$; and* pair *stores indices $[x, y]$ such that there is no pair with smaller distance*[2] *than (x, y) from among*

$$\left\{ (a, b) \,:\, \bigcup_{k=1}^{i} \{0, \ldots, k-1\} \times \{k, \ldots, n-1\} \right\}.$$

Lemma 2.. *Suppose that before the inner loops executes, $p = i$, for some $0 \leq i < n-1$ and $P(i)$ holds. Then, $Q(i, j)$: Suppose the inner loop executes at least j times. Then at the end of the j-th iteration $q = i + j + 1$; and pair stores indices $[x, y]$ such that there is no pair with smaller distance than (x, y) from among*

$$\left\{ (a, b) \,:\, \left(\bigcup_{k=1}^{i} \{0, \ldots, k-1\} \times \{k, \ldots, n-1\} \right) \cup \{i\} \times \{i+1, i+2, \ldots, i+j\} \right\}.$$

In addition, $min = \text{SQDIST}(L[x], L[y])$.

[2] Formally, a pair of indices i, j for which the distance between $L[i], L[j]$ is minimal.

Proof. $Q(i,0)$ holds trivially[3], from the initialization of q and the fact that $P(i)$ holds. Suppose $Q(i,j)$ holds and we shall prove $Q(i,j+1)$. Since $Q(i,j)$ holds, at the end of the j-th iteration of the inner loop *pair* stores indices $[x,y]$ such that (x,y) is a pair of indices of minimal distance from among

$$\left\{(a,b) \,:\, \left(\bigcup_{k=1}^{i}\{0,\ldots,k-1\}\times\{k,\ldots,n-1\}\right)\cup\{i\}\times\{i+1,i+2,\ldots,i+j\}\right\}.$$

Suppose the loop executes $j+1$ times. Then (from $Q(i,j)$) $q=i+j+1<n$ (by line 6), and by line 8 $d=\text{SQDIST}(L[p],L[q])$ (and we note that the indices are within the appropriate bounds). There are two options:

- If $d \geq min = \text{SQDIST}(L[x],L[y])$ (from $Q(i,j)$), then *pair* and *min* do not change and it is clear that (x,y) is a pair of indices of minimal distance from among

  $$\left\{(a,b) \,:\, \left(\bigcup_{k=1}^{i}\{0,\ldots,k-1\}\times\{k,\ldots,n-1\}\right)\cup\{i\}\times\{i+1,i+2,\ldots,i+j+1\}\right\}.$$

  Finally, q is incremented to $j+2$. We see that $Q(i,j+1)$ holds.

- If $d < min = \text{SQDIST}(L[x],L[y])$ (from $Q(i,j)$), then since *min* was the minimum distance pair (from $Q(i,j)$) of the set below, we have that d is smaller than the distance among any pair from

  $$\left\{(a,b) \,:\, \left(\bigcup_{k=1}^{i}\{0,\ldots,k-1\}\times\{k,\ldots,n-1\}\right)\cup\{i\}\times\{i+1,i+2,\ldots,i+j\}\right\}.$$

  That is, (p,q) is a pair of minimal distance from among

  $$\left\{(a,b) \,:\, \left(\bigcup_{k=1}^{i}\{0,\ldots,k-1\}\times\{k,\ldots,n-1\}\right)\cup\{i\}\times\{i+1,i+2,\ldots,i+j+1\}\right\}.$$

  Now, *pair* is changed to $[p,q]$ on line 11, and *min* is changed to d on line 12. Furthermore, q is incremented to $q+1$ on line 15. That is, $Q(i,j+1)$ holds.

$\square$

Proof of Lemma 1. $P(0)$ is just the initialization condition $p=0$. Suppose $P(i)$ holds and that the outer loop executes at least $i+1$ times. We shall prove $P(i+1)$ holds.

Since $P(i)$ holds we have $p=i$. Since the outer loop executes $i+1$ times, we must have $p=i<n-1$, by line 5. Then, q is initialized to $i+1$, and since $i+1<n$ the inner loop executes. By the end of the $i+1$-th iteration of the outer loop, the inner loop has also finished to execute, so we must have $q\geq n$ (the guard of the inner loop). Then, $p=i$ and the inner loop has executed at least $n-i-1$ times. Then by Lemma 2 $Q(i,n-i-1)$ holds. In particular, *pair* stores indices $[x,y]$ such that (x,y) is a pair of indices of minimal distance from among

$$\left\{(a,b) \,:\, \left(\bigcup_{k=1}^{i}\{0,\ldots,k-1\}\times\{k,\ldots,n-1\}\right)\cup\{i\}\times\{i+1,i+2,\ldots,n-1\}\right\}=$$

$$\left\{(a,b) \,:\, \bigcup_{k=1}^{i+1}\{0,\ldots,k-1\}\times\{k,\ldots,n-1\}\right\}$$

In addition, p increments to $p+1$ on line 13. That is, $P(i+1)$ holds. $\square$

[3] Note that $\{i\}\times\{i+1,\ldots,i+j\}$ is empty when $j=0$.

We are now ready to prove partial correctness.

Lemma 3.. *Suppose the precondition of the algorithm* MINDIST *are satisfied, and that it terminates. Then it returns a pair of points of L of minimal distance.*

Proof. Since $n \geq 2$, the outer loop is executed at least once. Since the algorithm terminates, the we must have $p \geq n-1$ on termination. Then the outer loop executes at least $n-1$ times. By Lemma 1, $P(n-1)$ holds and at the end of the $(n-1)$-th iteration pair stores indices $[x, y]$ such that (x, y) is a pair of indices of minimal distance from among

$$\left\{(a, b) \,:\, \bigcup_{k=1}^{n-1} \{0, \ldots, k-1\} \times \{k, \ldots, n-1\}\right\}.$$

Also, at the end of the $(n-1)$-th iteration, $p = n-1$ (again, by $P(n-1)$), so the loop will not iterate again. Then, line 14 will return $[L[x], L[y]]$, so it remains to show that these are the points of L of minimal distance.

We note that the distance function is symmetric $d(m, n) = d(n, m)$. Thus, saying that (x, y) is a pair of indices of minimal distance from among

$$\left\{(a, b) \,:\, \bigcup_{k=1}^{n-1} \{0, \ldots, k-1\} \times \{k, \ldots, n-1\}\right\}$$

is the same as saying that (x, y) is a pair of indices of minimal distance from among

$$\{(a, b) \,:\, \{0, \ldots, n-1\} \times \{0, \ldots, n-1\}\}$$

which is the same as saying that (x, y) is a pair of indices such that $L[x], L[y]$ are a pair of points of L of minimal distance. $\square$

It only remains to show that the algorithm terminates. There are several ways of doing so, we shall break the proof into two, as we did with the loop invariants.

Lemma 4.. *If the inner loop runs, it terminates.*

Proof. Let q_j denote the value of q at the beginning of each iteration of the inner loop. We shall show that $n - q_j$ is a decreasing sequence of natural numbers. We have $q_1 = p + 1 < (n-1) + 1 = n$ (since the inner loop executes, the outer loop's guard is met), so that $n - q_1 > 0$. Suppose the loop has executed $k \geq 1$ times, and is about to execute again. Then, $q_{k+1} = q_k + 1$ (e.g., by $Q(i, j)$). Since the loop executes again, $q_{k+1} < n$ and so $n - q_{k+1} > 0$. We have

$$n - q_{k+1} = n - q_k - 1 < n - q_k$$

which completes the proof. $\square$

Lemma 5.. *If the outer loop runs, it terminates.*

Proof. Since the inner loop always terminates, each run of the outer loop terminates (that is, results in an increment of p); and we just have to show that the loop itself terminates. This is done in the same way as with the previous lemma. Let p_j denotes the value of p at the beginning of each iteration of the outer loop. We shall show that $n - p_j$ is a decreasing sequence of natural numbers. We have $p_1 = 0$ so that $n - p_1 > 0$. Suppose the outer loop has executed $k \geq 1$ times, and is about to execute again. Then, $p_{k+1} = p_k + 1$ (e.g., by $P(k)$). Since the loop executes again, $p_{k+1} < n-1$ and so $n - p_{k+1} > 0$. We have

$$n - p_{k+1} = n - p_k - 1 < n - p_k$$

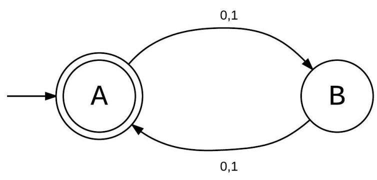
which completes the proof. $\square$

2. DFSAs and their operations

(a) Define and draw DFSAs on binary alphabet $\Sigma = \{0, 1\}$ for 2 languages: $L_1(M_1) = \{$all strings with even number of characters in a string$\}$, $L_2(M_2) = \{$all strings that have even number of 1s$\}$

Answer.

DFSA M_1:

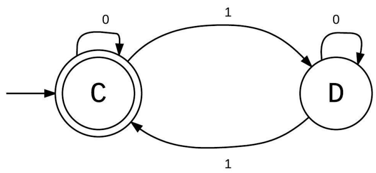


$\Sigma = 0, 1$
$Q = \{A, B\}$
Start State $= A$
Final/accepting states, $F = \{A\}$

$\delta =$

	0	1
A	B	B
B	A	A

DFSA M_2:



$\Sigma = 0, 1$
$Q = \{C, D\}$
Start State $= C$
Final/accepting states, $F = \{C\}$

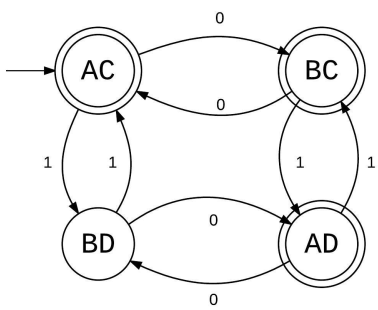
$\delta =$

	0	1
C	C	D
D	D	C

(b) Identify DFSA M_3 for the union of languages $L_1 \cup L_2$ - you can define it formally (don't need to draw).

Answer.

DFSA M_3 :



$\Sigma = 0, 1$
$Q = \{AC, AD, BC, BD\}$
Start State $= AC$
Final/accepting states, $F = \{AC, BC, AD\}$

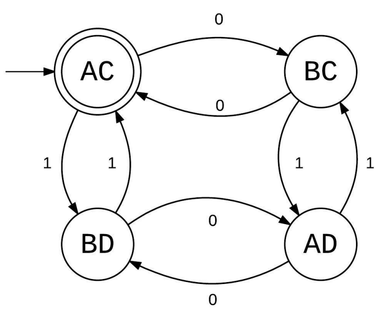
$\delta =$

	0	1
AC	BC	BD
AD	BD	BC
BC	AC	AD
BD	AD	AC

(c) Identify DFSA M_4 for the intersection of languages $L_1 \cap L_2$ - you can define it formally (don't need to draw)

Answer.

DFSA M_4 (Drawing the DFSA is optional):



$\Sigma = 0, 1$
$Q = \{AC, AD, BC, BD\}$
Start State $= AC$
Final/accepting states, $F = \{AC\}$

$\delta =$

	0	1
AC	BC	BD
AD	BD	BC
BC	AC	AD
BD	AD	AC

(d) Find and prove a state invariant for M_3.

Answer.

State invariant:

$$\delta^*(AC, s) = \begin{cases} AC & \text{if s has an even number of characters and an even number of 1s} \\ BC & \text{if s has an odd number of characters and an even number of 1s} \\ BD & \text{if s has an odd number of characters and an odd number of 1s} \\ AD & \text{if s has an even number of characters and an odd number of 1s} \end{cases}$$

Proof (by Structural Induction):
Let $P(s)$: The above state invariant is correct for the language $L_1 \cup L_2$, i.e. s has either an even number of characters or an even number of 1s.
Let Σ^* represent all the strings over the alphabet $\Sigma = \{0, 1\}$, including the empty string, i.e.

1. $\epsilon \in \Sigma^*$
2. if $x \in \Sigma^*$ then $x \circ 0, x \circ 1 \in \Sigma^*$

Basis:
$\delta^*(AC, \epsilon) = AC$ since ϵ is always in AC and according to the state invariant's first condition, has an even number of characters and an even number of 1's, which is the case. Therefore basis holds.

Induction Step:

Let $s' \in \Sigma^*$ and assume $P(s')$. We must show $P(s)$ where $s = s' \circ 0$ or $s = s' \circ 1$.

Case 1: $s = s' \circ 0$

$\delta^*(AC, s) = \delta^*(AC, s' \circ 0) = \delta(\delta^*(AC, s'), 0)$

$$\delta(\delta^*(AC, s'), 0) = \begin{cases} \delta(AC, 0) & \text{if s' has an even number of characters and an even number of 1s by IH} \\ \delta(BC, 0) & \text{if s' has an odd number of characters and an even number of 1s by IH} \\ \delta(BD, 0) & \text{if s' has an odd number of characters and an odd number of 1s by IH} \\ \delta(AD, 0) & \text{if s' has an even number of characters and an odd number of 1s by IH} \end{cases}$$

Further,

$$\delta(\delta^*(AC, s'), 0) = \begin{cases} BC & \text{if s has an odd number of characters and an even number of 1s} \\ AC & \text{if s has an even number of characters and an even number of 1s} \\ AD & \text{if s has an even number of characters and an odd number of 1s} \\ BD & \text{if s has an odd number of characters and an odd number of 1s} \end{cases}$$

The above is true since the number of 1s never changes when adding a zero and therefore we're just moving between states of odd/even number of characters by adding a zero.

<u>Case 2: $s = s' \circ 1$</u>
$\delta^*(AC, s) = \delta^*(AC, s' \circ 1) = \delta(\delta^*(AC, s'), 1)$

$$\delta(\delta^*(AC, s'), 1) = \begin{cases} \delta(AC, 1) & \text{if s' has an even number of characters and an even number of 1s by IH} \\ \delta(BC, 1) & \text{if s' has an odd number of characters and an even number of 1s by IH} \\ \delta(BD, 1) & \text{if s' has an odd number of characters and an odd number of 1s by IH} \\ \delta(AD, 1) & \text{if s' has an even number of characters and an odd number of 1s by IH} \end{cases}$$

Further,

$$\delta(\delta^*(AC, s'), 1) = \begin{cases} BD & \text{if s has an odd number of characters and an odd number of 1s} \\ AD & \text{if s has an even number of characters and an odd number of 1s} \\ AC & \text{if s has an even number of characters and an even number of 1s} \\ BC & \text{if s has an odd number of characters and an even number of 1s} \end{cases}$$

The above is true since the number of 1s is always increased by one and therefore we're moving between states of odd/even number of characters and odd/even number of ones by adding a one.

We've shown that $\delta^*(AC, s) \in \{AC, BC, AD\}$ (accepting states) if s has an even number of characters or an even number of 1s. We have also shown that if s has an odd number of characters and an odd number of 1s, s would be in the BD state. Therefore if s is not in the BD state, then s either has an even number of 1s or an even number of characters. More formally:
$\neg$(s has an odd number of characters and an odd number of 1s) $\Rightarrow \neg(\delta^*(AC, s) = BD)$
s has an even number of characters or an even number of 1s $\Rightarrow \delta^*(AC, s) \in \{AC, BC, AD\}$.
Therefore we have shown that $\delta^*(AC, s)$ is in an accepting state if and only if s has an even number of characters or an even number of 1s.

3. Language L over alphabet $\Sigma = \{a, b\}$ consists of all strings that start with a and have odd lengths or start with b and have even lengths:

$$\{s \mid s \text{ starts with } a \text{ and has odd length, or starts with } b \text{ and has even lengths}\}$$

(a) What is a regular expression R corresponding to language L?

$$R = \left(a\big((a+b)(a+b)\big)^* + b\big((a+b)(a+b)\big)^*(a+b)\right)$$

(b) Prove your regular expression R is indeed equivalent to L:

$To\ show\ L \equiv L(R), must\ show\ L(R) \subseteq L\ and\ L \subseteq L(R)$

$First, L(R) \subseteq L$:

$Note: L(R) = L\left(a\big((a+b)(a+b)\big)^* + b\big((a+b)(a+b)\big)^*(a+b)\right)$

$= L\left(a\big((a+b)(a+b)\big)^*\right) \cup L\left(b\big((a+b)(a+b)\big)^*(a+b)\right)$

$Case\ 1.\ s\ begins\ with\ an\ a\ and\ has\ an\ odd\ number\ of\ characters$

$s = L\left(a\big((a+b)(a+b)\big)^*\right)$

$s = L\left(a\big((a+b)(a+b)\big)^*\right) = L(a)L\left(\big((a+b)(a+b)\big)\right)^*$

$Let\ v \in L(a)\ and\ w \in L\left(\big((a+b)(a+b)\big)\right)$

$Then\ s = L\left(a\big((a+b)(a+b)\big)^*\right) = vw^*$

$Note\ that\ |v| = 1\ and\ |w^*| = 2k, k \in \mathbb{N}, k \geq 0$

$Then\ |vw^*| = 1 + 2k, k \in \mathbb{N}, k \geq 0$

$$\Rightarrow s = L\left(a\big((a+b)(a+b)\big)^*\right) \subseteq L$$

$Case\ 2.\ s\ begins\ with\ b\ and\ has\ an\ even\ number\ of\ characters$

$s = L\left(b\big((a+b)(a+b)\big)^*(a+b)\right)$

$s = L\left(b\big((a+b)(a+b)\big)^*(a+b)\right) = L(b)L\big((a+b)(a+b)\big)^*L(a+b)$

$Let\ x \in L(b), w \in L\big((a+b)(a+b)\big), z \in (a+b)$

$Then\ s = L\left(b\big((a+b)(a+b)\big)^*(a+b)\right) = zw^*z$

$Note\ that\ |x| = 1\ and\ |w^*| = 2k, k \in \mathbb{N}, k \geq 0\ and\ |z| = 1$

$Then\ |xw^*z| = 2 + 2k = 2k', k' = 1 + k, k \in \mathbb{N}, k \geq 0$

$$\Rightarrow s = L\left(b\big((a+b)(a+b)\big)^*(a+b)\right) \subseteq L$$

$Now, L \subseteq L(R)$:

$From\ the\ definition\ of\ L\ a\ string\ in\ L\ over\ the\ alphabet\ \Sigma = \{a, b\},$
$s = \{s \mid s\ starts\ with\ a\ and\ has\ odd\ length, or\ starts\ with\ b\ and\ has\ even\ lengths\}$

$Therefore\ there\ are\ two\ cases$:
$Case\ 1.\ \ The\ string\ s\ starts\ with\ a\ and\ has\ an\ odd\ number\ of\ characters$
$\Rightarrow s\ consists\ of\ first\ an\ a\ and\ then\ an\ even\ number\ of\ characters\ \in \Sigma$
$s = L\{x|x = a\}L\{y|y\ is\ an\ even\ number\ of\ characters \in \Sigma\}$
$Let\ s = vw$
$Where\ v \in L\{x|x = a\}\ and\ w \in L\{y|y\ is\ an\ even\ number\ of\ characters \in \Sigma\}$
$\Rightarrow v\ \in L(a)\ and\ w\ \in L\Big(\big((a+b)(a+b)\big)^*\Big)$
$\Rightarrow s = a((a+b)(a+b))^*$
$\Rightarrow L\{s\ |s\ starts\ with\ a\ and\ has\ odd\ length\} \in L(R)$

$Case\ 2.\ \ The\ string\ s\ starts\ with\ b\ and\ has\ an\ even\ number\ of\ characters$
$\Rightarrow s\ consists\ of\ b\ and\ an\ odd\ number\ of\ characters\ \in \Sigma$
$s = L\{x|x = a\}L\{y|y\ is\ an\ odd\ number\ of\ characters \in \Sigma\}$
$Let\ s = xt\ \ where\ x \in L\{x|x = b\}\ and\ t \in L\{x|x\ is\ an\ odd\ number\ of\ characters \in \Sigma\}$
$\Rightarrow x \in L(b)\ and\ t \in L\left((a+b)\big((a+b)(a+b)\big)^*\right)$
$\Rightarrow s = b(a+b)\big((a+b)(a+b)\big)^*$
$\Rightarrow L\{s\ |s\ starts\ with\ b\ and\ has\ even\ length\} \in L(R)$

$\Rightarrow L \subseteq L(R)$

$Therefore\ L(R) \subseteq L\ and\ L \subseteq L(R)\ \ \Rightarrow L \equiv L(R)$ ■