CSC236 Problem Set 2

Jacob Nazarenko, James Currier, Mark Abdullah

April 4, 2018

1. a)

$$T(n) = \begin{cases} \text{a} & \text{if } n = 1 \\ \text{b} + \frac{n}{2}\text{c} + 2T(\frac{n}{4}) & \text{if } n > 1 \end{cases}$$

Let a represent the constant time that the conditional and return statements on lines 1-2 take, and this occurs when $n = len(A) < 2$, therefore when $n = 1$ since $n \geq 1$.
The following occurs when $n \geq 2 \leftrightarrow n > 1$. Let b represent the time for the assignment and return statements on line 3 and 8, which assigns $len(A)//4$ to m. Let c represent the time for each iteration of the loop on lines 5-6. Since the loop iterates from the first quarter of A through the third quarter of A, it iterates $\frac{n}{2}$ times. Therefore $\frac{n}{2}$c represents the total time for the loop on lines 5-6. Lines 4 and 7 are recursive calls, each on a quarter of the list A, represented by $2T(\frac{n}{4})$.

b) We will apply the Master's Therom to determine an asymptotic upper-bound for $T(n)$.
We have $k = 1$, $d = 4$, and $c = 2$. Then by the Master's Therom,

$$\log_4 2 < 1 \implies T(n) \in \mathcal{O}(n)$$

2. a) *Algorithm*2 contains 2 loops, so we will define 2 loop invariants. Let LI-1 and LI-2 be the loop invariants for their respective loops.

Let $Q(A, k, e)$ represent the number of times e occurs in $A[0 : k]$. In order for us to have a majority element, we need to have $Q(A, k, e) > \lfloor \frac{len(A)}{2} \rfloor$.

Let **LI-1(k)** represent the assumption that if the loop executes k times, then

(1) $0 \leq m_k \leq i_k \leq len(A)$

(2) $Q(A, i_k, e_k) \geq m_k$

(3) $Q(A, i_k, e_k) - m_k \leq \lfloor \frac{i_k - m_k}{2} \rfloor$

(4) $\forall x \in A,\ x \neq e_k \implies Q(A, i_k, x) \leq \lfloor \frac{i_k - m_k}{2} \rfloor$

Let **LI-2(k)** represent the assumption that if the loop executes k times, then

(1) $0 \leq j_k \leq len(A)$

(2) $count_k = Q(A, j_k, e)$

Additionally, because there are 2 loop invariants in this algorithm, we must define 2 sets of pre and post-conditions. Let **Pre-1** and **Pre-2** be the preconditions, and let **Post-1** and **Post-2** be the post-conditions for their respective loops. We define them as follows:

Pre-1: A is a list
Post-1: If there exists a majority element x in list A, in other words if $\exists x \in A,\ Q(A, len(A), x) > \lfloor len(A)/2 \rfloor$, then $e = x$

Pre-2: A is a list, and e is defined with any given value
Post-2: If the value e occurs more than $\lfloor len(A)/2 \rfloor$ times in list A, then its value is returned; otherwise, nothing is returned

b) **Claim**: Precondition + LI-1 + Termination of 1st loop + LI-2 + Termination of 2nd loop $\implies$ Postcondition

Let us assume that the first loop terminates after t iterations. By the exit condition of the first loop, $i_t \geq len(A)$. We also know by (1) of LI-1 that $i_t \leq len(A)$. Therefore we may conclude that $i_t = len(A)$. We may now rearrange (3) from LI-1 to obtain the following relationship:

$$\begin{aligned} Q(A, len(A), e_t) - m_t &\leq \lfloor (len(A) - m_t)/2 \rfloor \\ 2 * Q(A, len(A), e_t) - 2 * m_t &\leq len(A) - m_t \\ 2 * Q(A, len(A), e_t) &\leq len(A) + m_t \\ Q(A, len(A), e_t) &\leq \lfloor (len(A) + m_t)/2 \rfloor \end{aligned}$$

Now, by (2) and (3) of LI-1, we have that $m_t \leq Q(A, len(A), e_t) \leq \lfloor \frac{len(A)+m_t}{2} \rfloor$. We also have by (4) of LI-1 that $\forall x \in A,\ x \neq e_t \implies Q(A, i_k, x_k) \leq \lfloor \frac{len(A)-m_t}{2} \rfloor$. To prove the post-condition holds, we can assume its hypothesis to be true, namely that A has a majority element z. We know from our new definition of the majority element that in this case, $Q(A, k, z) > \lfloor \frac{len(A)}{2} \rfloor$. Therefore, we may conclude that if the majority element e_t is not equal to this majority element z, then the above statement will contradict (4) of LI-1. We can therefore say that if A has a majority element, then e_t will in fact be equal to this element at the end, satisfying the post-condition of the first loop.

To satisfy the precondition of the second loop, e must be defined. We know that this holds, because of line 1 where m is set to 0, which causes line 3 to be executed at least once (during the first iteration of the loop). Line 3 defines e so we know it will definitely have a value when the second loop executes. Now let's assume that the second loop terminates after t iterations. By (1) of LI-2 and the exit condition of the loop, we know that $j_t \leq len(A)$ and that $j_t \geq len(A)$, so we can conclude that $j_t = len(A)$. By (2) of LI-2, we know that $count_t = Q(A, len(A), e)$. In lines 13-16, the algorithm uses our definition of the majority element above to determine whether or not to return the element e, so we may conclude that this definition, as well as by the outcome of the previous loop, that if e is the majority element, then the value of $count_t$ will be greater than $len(A)/2$ at return time, and therefore the value of e will be returned. If it is not, the nothing will be returned. This satisfies the post-condition of the second loop, which is the post condition of the entire algorithm.

c) **Claim:** $\forall n \in \mathbb{N}$, LI-1(n) holds

Proof (by simple induction):

Base case: $(n = 0)$
$i_0 = 0$
$m_0 = 0$
$e_0 = \text{None}$

$0 \le m_0 \le i_0 \le len(A) \implies (1)$ holds.
$0 = Q(A, i_0, e_0) \ge m_0 = 0 \implies (2)$ holds.
$0 - 0 = Q(A, i_0, e_0) - c_0 \le \lfloor \frac{i_0 - m_0}{2} \rfloor = 0 \implies (3)$ holds.
$\forall x \in A,\ 0 = Q(A, i_0, x) \le \lfloor \frac{i_0 - m_0}{2} \rfloor = 0 \implies (4)$ holds.

Therefore LI-1(0) holds.

Inductive Step: Let k $\in \mathbb{N}$ Assume LI-1(k) [IH]. WTP LI-1(k+1)
Since this loop contains an if, else if, else, we will split this proof into 3 paths.

Path 1: $m_k == 0$
By line 4, $m_{k+1} = 1$
By line 4, $e_{k+1} = A[i_k]$
Since this is a for loop, i increases by one every iteration, so $i_{k+1} = i_k + 1$ and $1 \le i_{k+1}$
Since the loop iterates $len(A)$ times, and m and i are increased at most 1 every iteration, initially equal to 0, $m_{k+1} \le i_{k+1} \le len(A)$

$$0 \le 1 = m_{k+1} \le i_{k+1} \le len(A) \implies (1) \text{ holds}$$

Since e_{k+1} occurs at least once in the $A[i_k]$ position,

$$Q(A, i_{k+1}, e_{k+1}) \ge 1 = m_{k+1} \implies (2) \text{ holds}$$

Since m_{k+1} is reassigned, we have 2 potential cases, $e_{k+1} = e_k$ and $e_{k+1} \ne e_k$. To prove (3) and (4), we will consider each case individually.

Case 1: $e_{k+1} = e_k$

$$\begin{aligned} Q(A, i_{k+1}, e_{k+1}) - m_{k+1} &= Q(A, i_k, e_k) + 1 - m_k - 1 \\ &= Q(A, i_k, e_k) - m_k \\ &\le \lfloor \frac{i_k - m_k}{2} \rfloor && \text{[IH] - (3)} \\ &= \lfloor \frac{i_k + 1 - m_k - 1}{2} \rfloor \\ &= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor \end{aligned}$$

Therefore we have shown that (3) holds.

Let $x \in A$. Assume $x \ne e_{k+1} = e_k$.

$$Q(A, i_{k+1}, x) = Q(A, i_k, x)$$

$$\leq \lfloor \frac{i_k - m_k}{2} \rfloor \qquad \text{[IH] - (4)}$$
$$= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor$$

Therefore we have shown that (4) holds.

Case 2: $e_{k+1} \neq e_k$

$$\begin{aligned} Q(A, i_{k+1}, e_{e+1}) - m_{k+1} &= Q(A, i_{k+1}, e_{k+1}) - 1 \\ &= Q(A, i_k, e_{k+1}) \\ &\leq \lfloor \frac{i_k - m_k}{2} \rfloor && \text{[IH] - (4)} \\ &= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor \end{aligned}$$

Therefore we have shown that (3) holds.

Now we want to show (4) holds. Let $x \in A$. Assume $x \neq e_{k+1}$.
Since $e_{k+1} \neq e_k$, we have 2 cases, $x = e_k$ and $x \neq e_k$. We will consider each case individually.

Case 2.1: $x = e_k$

$$\begin{aligned} Q(A, i_{k+1}, x) &= Q(A, i_{k+1}, e_k) \\ &= Q(A, i_k, e_k) \\ &= Q(A, i_k, e_k) - m_k && \text{(since } m_k = 0\text{)} \\ &\leq \lfloor \frac{i_k - m_k}{2} \rfloor && \text{[IH] - (4)} \\ &= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor \end{aligned}$$

Therefore we have shown that (4) holds.

Case 2.2: $x \neq e_k$
This is the same argument as in Case 1, refer to the details there. Therefore (4) holds.

Path 2: $A[i_k] = e_k$
Since e is not reassigned in this path, $e_{k+1} = e_k$
By line 6, $m_{k+1} = m_k + 1$
Since this is a for loop, i increases by one every iteration, $i_{k+1} = i_k + 1$
Since the loop iterates len(A) times, and m and i are increased at most 1 every iteration, initially equal to 0, $m_{k+1} \leq i_{k+1} \leq len(A)$

$$0 \leq m_k \leq m_{k+1} \leq i_{k+1} \leq len(A) \implies (1) \text{ holds}$$

$$Q(A, i_{k+1}, e_{k+1}) = Q(A, i_k, e_k) + 1 \qquad \text{A}[i_k] = e_k$$

$$\begin{aligned} &\geq m_k + 1 && \text{[IH] - (2)} \\ &= m_{k+1} \end{aligned}$$

Therefore we have shown that (2) holds.

$$\begin{aligned} Q(A, i_{k+1}, e_{k+1}) - m_{k+1} &= Q(A, i_k, e_k) + 1 - m_k - 1 \\ &= Q(A, i_k, e_k) - m_k \\ &\leq \lfloor \frac{i_k - m_k}{2} \rfloor && \text{[IH] - (3)} \\ &= \lfloor \frac{i_k + 1 - m_k - 1}{2} \rfloor \\ &= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor \end{aligned}$$

Therefore we have shown that (3) holds.

Let $x \in A$. Assume $x \neq e_{k+1} = e_k$.

$$\begin{aligned} Q(A, i_{k+1}, x) &= Q(A, i_k, x) \\ &\leq \lfloor \frac{i_k - m_k}{2} \rfloor && \text{[IH] - (4)} \\ &= \lfloor \frac{i_k + 1 - m_k - 1}{2} \rfloor \\ &= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor \end{aligned}$$

Therefore we have shown that (4) holds.

Path 3: $m_k \neq 0 \wedge A[i_k] \neq e_k$
Since e is not reassigned in this path, $e_{k+1} = e_k$
By line 8, $m_{k+1} = m_k - 1$
Since this is a for loop, i increases by one every iteration, $i_{k+1} = i_k + 1$
Since the loop iterates len(A) times, and m and i are increased at most 1 every iteration, initially equal to 0, $m_{k+1} \leq i_{k+1} \leq len(A)$

$$0 \leq m_k \wedge m_k \neq 0 \implies 1 \leq m_k \implies 0 \leq m_k - 1 = m_{k+1} \leq len(A) \implies (1) \text{ holds}$$

$$\begin{aligned} Q(A, i_{k+1}, e_{k+1}) &= Q(A, i_k, e_k) \\ &\geq m_k && \text{[IH] - (2)} \\ &\geq m_{k+1} \end{aligned}$$

Therefore we have shown that (2) holds.

$$
\begin{aligned}
Q(A, i_{k+1}, e_{k+1}) - m_{k+1} &= Q(A, i_k, e_k) - m_k + 1 \\
&\leq \lfloor \frac{i_k - m_k}{2} \rfloor + 1 && \text{[IH] - (3)} \\
&= \lfloor \frac{i_{k+1} - 1 - m_{k+1} - 1}{2} \rfloor + 1 \\
&= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor
\end{aligned}
$$

Therefore we have shown that (3) holds.

Let $x \in A$. Assume $x \neq e_{k+1} = e_k$.

$$
\begin{aligned}
Q(A, i_{k+1}, x) &\leq Q(A, i_k, x) + 1 && \text{(equality if } x = A[i_k]\text{)} \\
&\leq \lfloor \frac{i_k - m_k}{2} \rfloor + 1 && \text{[IH] - (4)} \\
&= \lfloor \frac{i_{k+1} - 1 - m_{k+1} - 1}{2} \rfloor + 1 \\
&= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor + 1 - 1 \\
&= \lfloor \frac{i_{k+1} - m_{k+1}}{2} \rfloor
\end{aligned}
$$

Therefore we have shown that (4) holds.

In all 3 Paths, we have shown all 4 conditions in LI-1 hold, therefore LI-1(k+1) holds.
Conclusion: By the Principle of Simple Induction, LI-1(k) holds $\forall k \in \mathbb{N}$. ■

We will now prove LI-2
Claim: $\forall k \in \mathbb{N}$, LI-2(k) holds

Proof (by simple induction):

Base case: ($k = 0$)
By line 9, $count_0 = 0$
$j_0 = 0$
$0 \leq j_0 = 0 \leq len(A) \implies$ (1) holds
Then, $Q(A, j_0, e_0) = 0 = count_0 \implies$ (2) holds.
Therefore LI-2(0) holds.

Inductive Step: Let n $\in \mathbb{N}$, Assume LI-2(n) [IH]. WTP LI-2(n+1).
Since there is only a single conditional check on line 11, we have 2 cases to consider, $A[j_k] = e_k$ and $A[j_k] \neq e_k$.

Case 1: $A[j_k] = e_k$

Because this is a for loop with $len(A)$ iterations where j is incremented by at most 1 at each iteration and starts off at 0, we can conclude that

$$0 \leq j_k \leq j_k + 1 = j_{k+1} \leq len(A) \implies (1) \text{ holds}$$

$$\begin{aligned} count_{k+1} &= count_k + 1 && \text{(line 12)} \\ &= Q(A, j_k, e) + 1 && \text{[IH] - (2)} \\ &= Q(A, j_{k+1}, e) \end{aligned}$$

Therefore we have shown (2) holds.

Case 2: $A[j_k] \neq e_k$
Once again, by the statement in the previous case regarding the value of j_{k+1}, (1) holds. Furthermore, the value of $count_{k+1}$ does not change from $count_k$ since no code is executed this iteration. The value of $Q(A, j_{k+1}, e)$ does not change since $A[j_k] \neq e_k$. Therefore by the assumption of [IH], (2) holds.

Since in both cases (1) and (2) hold, LI-2(k+1) holds.
Conclusion: By the Principle of Simple Induction, LI-1(n) holds $\forall n \in \mathbb{N}$. ■

3. a) Let **LI(k)** represent the assumption that if the loop executes k times, then

(1) $0 \leq i_k \leq end$

(2) $\forall p \in A[0 : i_k],\ EDist(A[end], p) \geq min_k$

(3) $min_k = EDist(p1_k, p2_k) \quad p1_k,\ p2_k \in A$

b) In order to prove the partial correctness of this algorithm, we must first define a loop measure, precondition, and postcondition for the while loop in lines 6-11:

Loop measure: Let m be the loop measure for this while loop, and let m_k be the value of this loop measure after k iterations. Let $m_k = end - i_k$. We must now prove that (a) $m_k \in \mathbb{N}$ and that (b) m_k is strictly decreasing.

(a) Case 1: When $k = 0$, $i = 0$ by line 5, so $end - i = end \implies m_k \in \mathbb{N}$. Case 2: When $k > 0$, we know by our LI and line 6 that $i \leq end$, so $m_k = end - i_k \geq 0 \implies m_k \in \mathbb{N}$.
(b) Notice that in the while loop that i_k is updated exactly once every iteration, on line 11, where i_k increases by 1, so we can say that

$$\begin{aligned} m_k &= end - i_k \\ &> end - i_k - 1 \\ &= end - (i_k + 1) \\ &= end - i_{k+1} && \text{(by line 11)} \\ &= m_{k+1} \end{aligned}$$

Thus we have proven that our loop measure is a natural number that is strictly decreasing with every iteration.

Precondition: A is a list of points in the Euclidean space, $0 < end < len(A)$, $len(A) > 1$, and min is defined as the minimum distance of all distances between all pairs of points in $A[0 \ldots end - 1]$.

Postcondition: If there exists a smaller distance between any of the points in $A[0 \ldots end-1]$ and $A[end]$ than the currently defined value of min, then this will be the the new value of min, and the corresponding points will be the new values of $p1$ and $p2$, upon termination of the loop.

Because we can see that the length of the list at each recursive call is exactly 1 smaller than the previous length, we may use simple induction to prove the correctness of this algorithm.

Proof (by simple induction):
Let $P(n)$ be defined as follows: If A is a list of points in the Euclidean space (each given as a pair of integers), $len(A) = n > 1$, and $0 < end < n$, then $Algorithm3$ will terminate and return a pair of points in $A[0 \ldots end]$ whose distance is the minimum of all the distances between all pairs of points in $A[0 \ldots end]$.

Claim: $\forall n \in \mathbb{N},\ n > 1,\ P(n)$ holds

Base case: $(n = 2)$
end can only be 1 in this case, so lines 1 and 2 will be executed, and the only two points in A will be returned. The distance between these points is the smallest because they are the only 2 points in A, and there can be no other combination of points. Therefore, $P(2)$ holds.

Inductive step:
Assume k is an arbitrary natural number greater than 1, and assume $P(k)$ holds [IH]. WTP that $P(k+1)$ holds.

If end is not equal to 1 (which we have already proved in our base case), then lines 3-12 will be executed. We know by our IH that the recursive call to $Algorithm3$ on line 3 will terminate and correctly return the pair of points in the sublist $A[0 : end]$ that have the smallest distance between them (which will be stored in min). The precondition to the loop in lines 6-11 will thereby be satisfied, so we may proceed by proving that this loop will terminate and return the correct result. By the loop exit condition and loop measure, we know that the loop terminates (after t iterations) when $m_t = end - i_t = 0$, so $i_t = end$. Then by (2) of our LI, we know that after the loop terminates $\forall p \in A[0 : i_t = end],\ EDist(A[end], p) \geq min_k$, So min_t is equal to the minimum distance between any two distinct points in $A[0 : end + 1]$. By (3) of our LI, the values of $p1_t$ and $p2_t$ are the values of the two points with the distance min_t. Therefore, when the loop terminates, the values of $p1_t$ and $p2_t$ must correspond to the two points with the smallest distance between them. Then in line 12, $p1_t$ and $p2_t$ are returned and the program ends. We have now shown that the post-condition for this algorithm will be reached for $len(A) = k + 1$.

We have thereby proven, by the Principle of Simple Induction, that $P(n)$ will hold for all $n \in \mathbb{N}$. ■

c) **Claim**: $\forall n \in \mathbb{N}$, LI(n) holds.

Proof (by simple induction):

Base case: $(n = 0)$
$i_0 = 0$
min_0 = minimum distance between any pair of points in the sublist $A[0 : end]$ (by assumption of the precondition)
$p1_0$ = 1st point from sublist $A[0 : end]$ in pair with minimum distance (by assumption of the precondition)

$p2_0$ = 2nd point from sublist $A[0 : end]$ in pair with minimum distance (by assumption of the precondition)
$end > 1$ due to the if-statement on lines 1-2 and the condition that $0 < end < len(A)$

$0 \leq i_0 = 0 \leq 1 < end \implies (1)$ holds

Since $i_k = 0, [0 : i_k]$ is the empty set
So $\forall p \in A[0 : i_k],\ EDist(A[end], p) \geq min_k \implies (2)$ holds

By the precondition min_0 is the minimum distance between any pair of points in the sublist $A[0 : end]$.
$p1_0$ and $p2_0$ are the points that have the minimum distance between them in the sublist $A[0 : end]$.
So $min_k = EDist(p1_k, p2_k) \implies (3)$ holds

Inductive Step: Let k $\in \mathbb{N}$ Assume LI(k) [IH]. WTP LI(k+1)

WTP (1) holds

Since i starts with the value 0, and it never decreases during the loop body, $i_{k+1} \geq 0$
By our Inductive Hypothesis, at the start of the loop $i_k < end$. Since i increases by at most one during an iteration of the loop, $i_k + 1 \leq end$
So $0 \leq i_{k+1} \leq end \implies (1)$ holds

WTP (2) holds

Case 1: EDist($A[end]$, i_{k+1})$\geq min_k$

Since the body of the if statement is not executed, $min_{k+1} = min_k$
By our inductive hypothesis, $\forall p \in A[0 : i_k],\ EDist(A[end], p) \geq min_k$
Since EDist($A[end]$, i_{k+1})$\geq min_k$
$\forall p \in A[0 : i_{k+1}],\ EDist(A[end], p) \geq min_k = min_{k+1} \implies (2)$ holds

Case 2: EDist($A[end]$, i_{k+1})$< min_k$
The body of the if statement is entered, so by line 8, min_{k+1} = EDist($A[end], A[i_{k+1}]$)
Since min_{k+1} = EDist($A[end]$, i_{k+1})$< min_k$
And by our Inductive Hypothesis $\forall p \in A[0 : i_k],\ EDist(A[end], p) \geq min_k$,
We get that $\forall p \in A[0 : i_{k+1}],\ EDist(A[end], p) \geq min_{k+1} \implies (2)$ holds

WTP (3) holds

Case 1: EDist($A[end]$, i_{k+1})$\geq min_k$
Then the body of the if statement will not be executed and the values of $min_{k+1}, p1_{k+1}$, and $p2_{k+1}$ are not reassigned.
So, $min_{k+1} = min_k, p1_{k+1} = p1_k$, and $p2_{k+1} = p2_k$
Then, by our Inductive Hypothesis that $min_k = EDist(p1_k, p2_k) \quad p1_k,\ p2_k \in A$,

$min_{k+1} = EDist(p1_{k+1}, p2_{k+1}) \quad p1_{k+1},\ p2_{k+1} \in A \implies (3)$ holds

Case 2: EDist($A[end]$, i_{k+1})$< min_k$
Then the body of the if statement on line 7 will be executed, so
by line 8, $min_{k+1} =$ EDist$(A[end], A[i_{k+1}])$
by line 9, $p1_{k+1} = A[end]$
by line 10, $p2_{k+1} = A[i_{k+1}]$
So $min_{k+1} = EDist(p1_{k+1}, p2_{k+1}) \quad p1_{k+1},\ p2_{k+1} \in A \implies (3)$ holds

We have thereby proven, by the Principle of Simple Induction, that LI(n) holds $\forall n \in \mathbb{N}$. ■