1 (8 points) \( a \) Let \( T \) be an unsorted array of \( n \) integers. Give an algorithm to find a pair \( x, y \in T \) which maximizes \(|x - y|\). If \( T = (6, 13, 19, 3, 8) \), then \( x = 19 \) and \( y = 3 \) would be a solution. The worst case execution time of your algorithm must be in \( O(n) \).

\( b \) Let \( T \) be a sorted array of \( n \) integers. Give an algorithm to find a pair \( x, y \in T \) which maximizes \(|x - y|\). If \( T = (3, 6, 8, 13, 19) \), then \( x = 3 \) and \( y = 19 \) would be a solution. The worst case execution time of your algorithm must be in \( O(1) \).

\( c \) Let \( T \) be an unsorted array of \( n \) integers. Give an algorithm to find a number \( x \) which doesn’t appear in \( T \). If \( T = (6, 13, 19, 3, 8) \), then \( x = 5 \) be a solution. The worst case execution time of your algorithm must be in \( O(n) \).

\( d \) Let \( T \) be a sorted array of \( n \) integers. Give an algorithm to find a pair \( x, y \in T \) which minimizes \(|x - y|\). If \( T = (3, 6, 8, 13, 19) \), then \( x = 6 \) and \( y = 8 \) would be a solution. The worst case execution time of your algorithm must be in \( O(n) \).

1. \( a \) We can find MAX and MIN in worst case linear time (for example, use \textit{maxindex} on page 424 of our text). Then just set \( x \leftarrow \text{MAX} \) and \( y \leftarrow \text{MIN} \).

\( b \)
\[
\begin{align*}
x & \leftarrow T[1] \\
y & \leftarrow T[n]
\end{align*}
\]

\( c \)
\[
\begin{align*}
x & \leftarrow 1 \\
\text{for } i & \leftarrow 1 \text{ to } n \text{ do} \\
& \quad x \leftarrow x + |T[i]|
\end{align*}
\]

That is, \( x \leftarrow 1 + \sum_{i=1}^{n} |T[i]| \).

\( d \)
\[
\begin{align*}
\text{DIFF} & \leftarrow \infty \\
\text{for } i & \leftarrow 2 \text{ to } n \text{ do} \\
& \quad \text{if } (\text{DIFF} > |T[i] - T[i-1]|) \text{ then } \text{INDEX} \leftarrow i \\
x & \leftarrow T[\text{INDEX}-1] \\
y & \leftarrow T[\text{INDEX}]
\end{align*}
\]
2. (4 points) From Baase and Van Gelder’s *Computer Algorithms*

Suppose an algorithm does \( m^2 \) steps on an array of \( m \) elements (for any \( m \geq 1 \)). The algorithm is to be used on two arrays \( A_1 \) and \( A_2 \) (separately). The arrays contain a total of \( n \) elements. \( A_1 \) has \( k \) elements and \( A_2 \) has \( n-k \) elements \((0 \leq k \leq n)\).

For what value(s) of \( k \) will the most work be done? For what value(s) of \( k \) will the least work be done? Justify your answers. (Remember that an example is not a proof. There is a good solution for this problem using simple calculus.)

The work done is 

\[
f(k) = k^2 + (n-k)^2 = 2k^2 - 2nk + n^2,
\]

which is the equation of a parabola. To find the extreme values of \( f(k) \), we differentiate with respect to \( k \) and set it equal to 0, yielding 

\[
\frac{d}{dk} f(k) = 4k - 2n = 0.
\]

The value \( k = \frac{n}{2} \) minimizes \( f(k) \) (because the second derivative with respect to \( k \) is 4, which is positive). On each side of the minimum, the sign of \( f'(k) \) doesn’t change. Thus, the maxima will be at the endpoints of the interval. Since \( f(k) \) is equal at the endpoints, it achieves its maximum value for \( 0 \leq k \leq n \) at the endpoints of \( k=0 \) and \( k=n \). The minimum value is \( \frac{n^2}{2} \) and the maximum value (at both endpoints) is \( n^2 \).

3. (16 points) Write programs to implement a (min)-priority queue using each of the following data structures:

- Ordered array
- Unordered array
- Binary search tree
- Heap

Actually, you only need programs to implement `construct` and `insert`. For each implementation, estimate the average time to insert \( n \) elements into an empty priority queue. Describe the implementation you use, and show results supporting your estimate.