1. First the median between 15, 8, and 3 is computed, and swapped into the last element of the subarray, leading to the array: \langle 15, 7, 9, 3, 4, 10, 8 \rangle. So 8 will be the pivot element. Next 15 (the first element moving forwards from the left ≥ 8) and 4 (the first element moving backwards from the right < 8) are swapped giving the array: \langle 4, 7, 9, 3, 15, 10, 8 \rangle. Next 9 (the next element moving forwards from the left ≥ 8) and 3 (the next element moving backwards from the right < 8) are swapped giving the array: \langle 4, 7, 3, 9, 15, 10, 8 \rangle. Finally, the pivot element 8 is swapped with 9 (the leftmost element ≥ 8) to lead to the partitioned array \langle 4, 7, 3, 8, 15, 10, 9 \rangle.

2. Suppose that on average each transaction is out of its proper sorted position by at most \( c \) positions for some constant \( c \). This means that over all \( n - 1 \) iterations of insertion sort, the number of comparisons is at most \( (c + 1)n \) (with one final comparison used to recognize the element is in order) and at most \( cn \) swaps are made. This gives a running time of \( \Theta(cn) = \Theta(n) \).

Thus, insertion sort is the best choice since mergesort has worst case \( \Theta(n \log n) \) time complexity, and quicksort has expected case time complexity \( \Theta(n \log n) \).

3. Let the random variable \( X \) be the index of the minimum element. \( X \) takes on value \( i \) (for \( i \in \{0, 1, \ldots, n - 1\} \)) with probability \( 1/n \). Thus

\[
E[X] = \sum_{i=0}^{n-1} i \cdot \text{Probability}(X = i) = \sum_{i=0}^{n-1} \frac{i}{n} = \frac{1}{n} (0 + 1 + \cdots + n - 1) = \frac{n(n - 1)}{2n} = \frac{n - 1}{2}.
\]

4. Let the random variable \( X \) be the number of times that \( A[i] \neq x \) is executed.

(a) Here \( X \) can only take on one of 3 values: 1 (occurs with probability \( 1/2 \) when \( x \) is in \( A[0] \)), 2 (occurs with probability \( 1/4 \) when \( x \) is in \( A[1] \)) and \( n \) (occurs with probability \( 1/4 \) when \( x \) is not in \( A \)). Thus

\[
E[X] = 1/2 \cdot 1 + 1/4 \cdot 2 + 1/4 \cdot n = n/4 + 1
\]

(b) Here \( X \) can take on values between 1 and \( n \) each with probability of \( 1/(2n) \) when the search is successful and with probability of \( 1/2 \) it takes on a value of \( n \) (an unsuccessful search). So

\[
E[X] = \left[ \sum_{i=1}^{n} \frac{i}{2n} \right] + n \cdot \frac{1}{2} = \frac{1}{2n} \left( 1 + 2 + \cdots + n \right) + n/2 = \frac{1}{2n} \frac{n(n+1)}{2} + \frac{n}{2} = \frac{3n}{4} + 1/4
\]

(c) Here when the search is successful, \( X \) takes on value \( i \) with probability \( (1/2)^i \) for \( i = 1, \ldots, n \) and when the search is unsuccessful, \( X \) takes on value \( n \) (and this occurs with probability \( 1/(2^n) \)). Thus

\[
E[X] = \left[ \sum_{i=0}^{n} \frac{i}{2^n} (1/2)^i \right] + n \cdot \frac{1}{2^n} = 2 - \frac{1}{2^{n-1}} - \frac{n}{2^n} + \frac{n}{2^n} = 2 - \frac{1}{2^{n-1}}
\]
5. Let the random variable $X$ be the number of times that `ptr.value != x` is executed.

We have that:

<table>
<thead>
<tr>
<th>position of item searched for</th>
<th>number times <code>ptr.value != x</code> done</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1/4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$i$</td>
<td>$i$</td>
<td>1/(2$^i$)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$n-1$</td>
<td>$n-1$</td>
<td>1/(2$^{n-1}$)</td>
</tr>
<tr>
<td>$n$</td>
<td>$n$</td>
<td>1/(2$^{n-1}$)</td>
</tr>
</tbody>
</table>

So expected number of times `ptr.value != x` is executed is \( \left( \sum_{i=1}^{n-1} i/(2^i) \right) + n/(2^{n-1}) \).

Applying formula that \( \sum_{i=1}^{k} \frac{i}{2^i} = 2 - \frac{1}{2^{k-1}} - \frac{k}{2^k} \) with \( k = n - 1 \) yields that expected number of times `ptr.value != x` is executed is:

\[
2 - \frac{1}{2^{n-2}} - \frac{n - 1}{2^{n-1}} + \frac{n}{2^{n-1}} = 2 - \frac{2}{2^{n-1}} + \frac{1}{2^{n-1}} = 2 - \frac{1}{2^{n-1}}
\]

So you on average you the number of list items traversed approaches 2 as \( n \) approaches infinity.