1. The following is the pseudo-code for a sorting procedure known as Bubble sort for sorting an array of \( n \) integers in ascending order.

   ```
   for (i = 0; i < n-1; i++) {
       swapped := false;
       for (j = n-1; j > i; j--)
           if A[j-1] > A[j] then swap them and set swapped to true;
       if swapped is false then halt;
   }
   ```

   (a) Run bubble sort on the sequence 10 12 8 9 5 7.
   (b) Derive the time complexity of bubble sort.
   (c) What is the worst-case input for bubble sort? Use it to derive a lower bound on the time complexity of bubble sort in the worst case.
   (d) What is the best-case input for bubble sort? What is the time complexity of bubble sort for sorting this best-case input?

2. Answer the following questions with explanations.

   (a) Let A and B be two algorithms for the same problem. If A runs in \( O(n^4) \) time and B runs in \( O(n^2) \) time, then is B is a more efficient algorithm than A?
   (b) Let A and B be two algorithms for the same problem. If A runs in \( \Omega(n^4) \) time and B runs in \( \Omega(n^2) \) time, then is B is a more efficient algorithm than A?
   (c) Let A and B be two algorithms for the same problem. If A runs in \( \Theta(n^2) \) time and B also runs in \( \Theta(n^2) \) time, then will A and B take exactly the same absolute time for every input?

3. Describe the mergesort algorithm in plain English. No pseudocode is allowed.

4. Insert the sequence of numbers 10 12 8 9 5 7 1 2 6 3 in this order into an initially empty binary search tree. Draw each intermediate tree obtained.