Problem 1. One way to find the median of a list is to sort the list and then take the middle element.

1. Assume you use Bubble sort to sort a list with 7 elements (i.e., \( n = 7 \)). Exactly how many comparisons do you use (in the worst case)?

2. Assume you use Mergesort to sort a list with 7 elements (i.e. \( n = 7 \)). Exactly how many comparisons do you use (in the worst case)?

Problem 2. You can actually find the median by running a sorting algorithm and stopping early, as soon as you know the median.

1. Assume you use Bubble Sort to find the median of 7 elements (i.e. \( n = 7 \)), but stop as soon as you know the median. Exactly how many comparisons do you use (in the worst case)?

2. Assume you use Mergesort to find the median of 7 elements (i.e. \( n = 7 \)), but stop as soon as you know the median. Exactly how many comparisons do you use (in the worst case)?

Problem 3. We are given \( n \) points in the unit circle, \( p_i = (x_i, y_i) \), such that \( 0 < x_i^2 + y_i^2 \leq 1 \) for \( i = 1, 2, \ldots, n \). Suppose that the points are uniformly distributed; that is, the probability of finding a point in any region of the circle is proportional to the area of that region. Design an algorithm with an average-case running time of \( \Theta(n) \) to sort the \( n \) points by their distances \( d_i = \sqrt{x_i^2 + y_i^2} \) from the origin.