1. A prune-and-search algorithm consists of several iterations. At each iteration, it prunes away a fraction \( f, 0 < f < 1 \), of input data, and then it invokes the same algorithm recursively to solve the problem for the remaining data. After a certain number of iterations, the size of input data will be so small that the problem can be solved directly with constant time. Assume that the time complexity of a prune-and-search algorithm is \( T(n) \) and the time needed to execute each iteration is \( O(n^k) \) for some constant \( k, k>0 \). Show that \( T(n)=O(n^k) \). (20%)

2. Show that the average case lower bound of the sorting problem is \( \Omega(n \log n) \). (15%)

3. Given a problem \( X \), how can we prove it to be NP-hard? (15%)

4. Given a list of \( n \) positive integers, you are asked to partition the list into two sublists, each of size \( \lfloor n/2 \rfloor \) or \( \lceil n/2 \rceil \), such that the difference between the sums of the integers in the two sublists is minimized. (a) Give a decision version of this problem, and show that it is NP-complete. (15%)
   (b) If the summation of these \( n \) positive integers is small enough, it is possible to solve this problem efficiently. Design such an algorithm. (15%)

5. For two points \( P = (p_1, p_2) \), and \( Q = (q_1, q_2) \) in the plane, we say that \( P \) dominates \( Q \) if \( p_1 > q_1 \) and \( p_2 > q_2 \). Given a set \( S \) of \( n \) points, the rank of a point \( P \) in \( S \) is the number of points in \( S \) dominated by \( P \). The problem is to find the rank of every point in \( S \). A straightforward way to solve this problem is to conduct an exhaustive comparison of all pairs of points. Hence, this approach requires \( O(n^2) \) running time. Design a faster algorithm (i.e. the running time of your algorithm must be in \( o(n^2) \) order) to solve this problem. (20%)