Problem 1: Back in Numberia

After finding and replacing the missing bag of twigs, you are treated as a hero in NUmberia. The thieves are so upset at your solution, that they break into the shrine and rearrange all of the bags of twigs, so they are no longer sorted. After your recent success, the Numberians rush to you, to aid in sorting the bags.

1. Given any two bags of twigs, what is the smallest number of steps to determine which bag has more twigs? (express your answer as a function of the number of twigs in each bag)

2. Given the number obtained in (1), how many steps are required to perform randomized quicksort?

Problem 2: Maximum Subsequence sum

The maximum subsequence sum problem is defined as follows: Given an array $A[1...n]$ of positive and negative integers, find indices $i$ and $j$ with $1 \leq i \leq j \leq n$ such that

$$\sum_{k=i}^{j} A[k]$$

is maximized. For example if $A = [2, -1, 5, -4, 1]$, then the maximum sum is 6, which is given by index 1 through 3.

1. Give a divide and conquer algorithm which solves this problem in time $O(n \log n)$. (Hint: the recursion is similar to that of MergeSort)

2. (harder) Give an iterative algorithm which solves this problem in time $O(n)$.

Problem 3: Find the sum

You are given as input a set $S$ of $n$ real numbers and a real number $x$. ($S$ is represented as an array).

1. Design an algorithm to determine whether there are two elements of $S$ whose sum is exactly $x$. Your algorithm should run in time $O(n \log n)$.

2. Suppose that the set $S$ is given in sorted order. Design an algorithm to solve this problem in time $O(n)$.  

---

1 You may work with a partner, but you must submit your own solutions. You may NOT consult any textbook other than CLRS or the internet to aid in solving these problems.
Problem 4: Find k’th largest element of a union

As input, you are given two sorted lists \( A \) and \( B \), both containing \( n \) elements. For this problem assume that all the elements of \( A \) and \( B \) are distinct. It is trivial to find the \( k \)'th largest element of \( A \) in time \( O(1) \) by simply outputting \( A[k] \). Similarly, we can find the \( k’th \) smallest element of \( B \). Your job is to design an algorithm which find the \( k \)'th largest element in \( A \) union \( B \). Your algorithm should run in time \( O(\log n) \). (A trivial solution would be to merge \( A \) and \( B \) into a new sorted array \( C \), then output \( C[k] \). However, this runs in time \( O(n) \), which does not meet the time requirement.) Hint: Consider modifying binary search.

Problem 5: Nuts and Bolts

You are given a collection of \( n \) bolts of different widths and \( n \) corresponding nuts. Each bolt matches 1 and only 1 specific nut. You are allowed to try a nut and a bolt together, from which you can determine whether the nut is too large, too small, or an exact match for the bolt. However, there is no way to compare two nuts together or two bolts together. The goal is to match each bolt to its nut.

1. Devise an algorithm to solve the nuts and bolts problem in expected time \( O(n \log n) \).

2. (harder) Show that any algorithm which solves the nuts and bolts problem must take \( \Omega(n \log n) \) comparisons in the worst case. Hint: Modify the proof of lower bounds for sorting to fit this problem.