[50] Homework 5. *Binary Search and Sorting*

Due by: November 13 by the end of the class.

[10] Let $A[1..n]$ be a sorted array of *distinct* integers. Give a divide-and-conquer algorithm that finds an index i such that $A[i] = i$ (if it exists) and runs in time $O(\log n)$.

[10] The input set S contain n real numbers. Let x be given.

(a) Design an algorithm that finds (if exist) two elements of S whose sum is x. The algorithm should run in time $O(n \log n)$.

(b) Suppose now S is given in a sorted order. Find an algorithm that solves the above problem in $O(n)$ time.

[10] Assume an array $A[1:n]$ is given. We know that after sorting every element originally at position i will end up at the final position $P[i]$ such that

$$|P[i] - i| \leq \log^3 \log n.$$

Design and efficient algorithm to sort $A[1:n]$. Then, establish the complexity of the optimal algorithm. Make sure your algorithm designed above is optimal.

[10] Let a set of n real numbers, say $a_1, \ldots, a_n$, be given. Assume n is even. Next, we partition the set into $n/2$ pairs, and then for every pair we compute the sum of its numbers. Thus, after such a partition we have $n/2$ sums $s_1, \ldots, s_{n/2}$. Propose an algorithms running in $O(n \log n)$ time that finds the partition minimizing the *maximum* sum. You *must* prove that your algorithm is correct.

[10] The input is a max-heap of size n (given as an array), and a real number x. Design an algorithm to determine whether the kth largest element in the heap is less than or equal to x. The worst-case running time of your algorithm must be $O(k)$ independent of the size of the heap. Justify your answer!
(**Hint**: Notice that you do **not** need to find the kth largest element; you need only to determine its relationship to x.)