Problem set 7.

Heapsort, Binsort, Radixsort.

1. (a) Construct a heap using the linear time BUILD-HEAP method from the following array of numbers: $31, 6, 50, 7, 2, 51$.
   (b) Insert the numbers 1 and then 5 into the resulting heap.
   (c) Perform two consecutive DELETEMIN operations on the resulting heap.
2. Sort the array $7, 3, 15, 1, 5, 4, 8, 2$ using (a) Binsort (b) Heapsort.
3. Sort the following list using Radixsort: abc, acb, bca, bbc, acc, bac, baa.
4. The array $A[1 : n]$ stores integers (a number may repeat several times). Define an algorithm with running time $O(n \log n)$ that determines a value that occurs more than once in the array.

---

5. (a) Give an efficient algorithm for finding the second smallest element in a heap.
   (b) Give an efficient algorithm for finding the tenth smallest element in a heap.
6. Given the position of an element in a Heap, show that one can modify its value (increase or decrease) while maintaining the Heap property in $O(\log n)$ steps.
7. Given an array $n \geq 2$ of distinct numbers, we want to find the pair whose difference is minimal (i.e. find the closest pair of numbers). Give a comparison based algorithm with running time $O(n \log n)$ for this task.
8. Let $A[1 : n]$ be an array of integers and let b also be an integer. We want to determine if there are indices $i, j \in \{1, \ldots, n\}$ for which $A[i] + A[j] = b$. Please solve this problem with a running time of $O(n \log n)$.
9. We are given two arrays, each containing n distinct integers. Give an algorithm with running time $O(n \log n)$ to find the smallest common element of the two arrays.
10. Dr. Watson states to Sherlock Holmes that he knows a comparison-based sorting algorithm that sorts an array of any size in such a way that it is guaranteed to compare each element of the array at most 2023 times. How can Sherlock Holmes convince Watson that his algorithm must be incorrect?

---

11. We are given an array of n not necessarily distinct integers. Our task is to design an algorithm with running time $O(n \log n)$ that determines the modes (most frequent elements) of the array, that is, the elements which occur the maximum number of times in the array.
12. We are given the adjacency list of a simple undirected graph. Give an algorithm that determines the most frequent degree in the degree sequence of the graph. The running time of the algorithm should be $O(n + m)$.
13. We are given an array $A[1 : n]$ with all distinct integers. We want to make an array $B[1 : n]$ from the elements of A such that the values alternate up and down, i.e. $B[1] < B[2] > B[3] < B[4] > \ldots$.
    (a) Give an algorithm with running time $O(n \log n)$ for this problem.
    (b) Give an algorithm with running time $O(n)$ for this problem.
14. We are given an array A of n distinct numbers and another number $1 \leq k \leq n$. We want to determine k numbers with the smallest absolute value from the array. If there are several such solutions, then it is sufficient to specify only one. Give an algorithm that determines k such values with a running time of $O(n)$ when $k \leq \lfloor \log n \rfloor$.
15. Given an array A containing n distinct integers, we want to find three numbers such that the difference of any two of them is at most 2019. Give an algorithm with running time $O(n \log n)$ for this task.

16. We are given two words over an alphabet I comprising of 4 letters: $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_k$, where $1 \leq k \leq n$ and $x_i, y_j \in I$. We want to find the subwords in x that are anagrams of y, i.e., we want to find all indices i such that $x_i, x_{i+1}, \ldots, x_{i+k-1}$ can be rearranged to give the word y. Give an algorithm that finds all such indices i in x with a running time of $O(n)$.

17. $A[1:n]$ is a sorted array of numbers given in increasing order. Unfortunately, before the array is given to us, k of its elements are replaced by someone. The locations of the substituted k elements are are not known, but we do know the number k. Can you give an algorithm with running time $O(n + k \log k)$ for sorting this array?

18. We are given a matrix of size $n \times n$. Give a comparison-based algorithm with running time $O(n^2 \log n)$, that decides whether there are two rows whose elements in the first column are different, but whose elements in all other columns are the same.