5. Sorting

Problem 5.1

Assign each of the following algorithms to the appropriate column(s): insertion sort, selection sort, bubble sort, merge sort, heap sort, quicksort, bucket sort, counting sort and radix sort.

Comparison-based:
Non comparison-based:
Worst-case time complexity $O(n^2)$:
Worst-case time complexity $O(n \log n)$:
In-place:
Stable:

Problem 5.2

Consider the following array where keys are integers and data is of type string:

(5, B)	(3, A)	(2, B)	(3, C)	(1, B)	(2, D)	(2, A)

Give the result of sorting this array in increasing order using the following algorithms: insertion sort, quicksort and heap sort.

Problem 5.3

Trace the execution of radix to sort the following array in increasing order. Use 10 as base and give the content of the array after sorting with each digit.

125	601	25	7	108	55	402

Problem 5.4

Consider the following variation of the the counting sort algorithm seen in class:

```
public static void countingSort(int[] A, int n, int m) {
    int[] counts = new int[m];
    for (int j = 0; j < m; j++)
        counts[j] = 0;
    for (int i = 0; i < n; i++)
        counts[A[i]]++;
    for (int j = 1; j < m; j++)
        counts[j] += counts[j - 1];
    int[] B = new int[n];
    for (int i = 0; i < n; i++) { // Change is made here
        B[counts[A[i]] - 1] = A[i];
        counts[A[i]]--;
    }
    for (int i = 0; i < n; i++)
        A[i] = B[i];
}
```

We changed the direction of the 4th for loop from descending to ascending. Discuss the properties of this version:

- Time complexity.
- Space complexity.
- In-place?
- Stable?

Problem 5.5

Discuss the properties (time and space complexity and stability) of the sorting algorithms seen in class when the elements to be sorted are stored in a linked list. Which algorithms retains the same time performance?

Problem 5.6

Give an implementation of merge sort where the array is divided into three parts instead of two. What is the time complexity of this version? Compare it to the version seen in class.

Problem 5.7

Write an implementation of quicksort where the pivot is selected randomly.

Problem 5.8

The implementation of radix sort seen in class uses division to extract digits. This approach has the advantage that any (strictly positive) integer can be used as base but is not efficient due to the computational cost of the division operation. Write an implementation of radix sort where the base is always a power of 2: the method takes as input the power as integer ($log_2(b)$) instead of the base itself. Write this method without using division (hint: use bit shift).

Problem 5.9

We cannot sort elements using two keys at the same time, but we can use one key as primary key and the other as a secondary key: we first sort according to the primary key then sort elements with equal primary keys using the secondary key. This procedure can obviously be generalized to multiple keys.

Write the method `void sort(int[][] A, int[] colIndex)` that sorts the rows of `A` using the columns specified in `colIndex`.

■ **Example 5.1** Consider the following two dimensional array:

```
2   5   3   2
4   3   3   3
5   2   2   2
4   4   3   1
3   2   1   1
2   4   3   5
2   5   2   1
1   5   1   4
2   4   4   3
3   5   5   5
```

Sorting this array according to column 1, then 0, then 3 gives:

```
3   2   1   1
5   2   2   2
4   3   3   3
2   4   4   3
2   4   3   5
4   4   3   1
1   5   1   4
2   5   2   1
2   5   3   2
3   5   5   5
```

■