Data Structures, exercises week 10, 12-16.4

1. In the following problems the input of the algorithm is an array T that contains n integer values. Give an solution algorithm to each problem that works in time $O(n \log n)$.

   (a) Does some value occur multiple times in the array?

   (b) Which value occurs the most in the array?

   (c) What is the smallest difference between two values in the array?

2. An algorithm gets as input an array T which contains n integers and an integer k. Give an solution algorithm working in time $O(n \log n)$, which checks if the array has two values, which produce sum k? Additional challenge (not required for the cross):

   (a) Give an algorithm working in time $O(n^2)$, which checks if the array has three values, which produce sum k

   (b) Give an algorithm working in time $O(n^2 \log n)$ which checks if the array has four values, which produce sum k

   Let's assume, that the one position in array T[i] can't belong to the sum more than once.

3. Implement mergesort or quicksort (or rather both) with Java or any programming language.

   Java API `Arrays` class has a method `sort` that enables sorting an array of any type.

   Measure the performance of the sorting algorithms you did at this and last week with Javan Arrays.sort. Do the measurements such that the input is

   - in random order
   - ready in order
   - ready in inverted order

   If you are using some other programming langueage than Java, take as comparision point the sort algorithm implementation in the standard of your language.

4. You can sort an array in time $O(n \log n)$. But how fast can you shuffle an array? Define a efficient algorithm for shuffling an array. What is the time and the space complexity of your algorith? You can assume, that you have a funcion, that returns an integer between $1 \ldots n$ in constant time. Does your algorithm shuffle the array so that all the orders of the values are equally probable?

5. The *transpose* G^{T} of a directed graph $G = (V, E)$ is obtained by changing the direction of every edge. In other words, $G^{\mathrm{T}} = (V, E^{\mathrm{T}})$, where $E^{\mathrm{T}} = \{(v, u) \mid (u, v) \in E\}$. Give an algorithm that given G creates G^{T} in time $O(|V| + |E|)$ when the graphs are represented using adjacency lists.

6. Implement a program the finds the shortest way out of a labyrinth. You can use any programming language, or if you do not want to do programming, a fairly precise pseudo code is enough.

   The program is given as input a labyrinth, where `#` means the wall, `.` means the floor and `X` means the starting point. The program must print again the labyrinth so that the shortest way out is shown. If there is no way out, the program must report it. You can get out of the labyrinth from all the locations, where there is floor on the border.

   If you give this as input to the program:

```
####.##.#
...#....#
#.#.###.#
#....X..#
#########
```

   Then this is the output of the program:

```
####.##o#
...#...o#
#.#.###o#
#....Xoo#
#########
```

   The shortest way out is 5 steps, which are marked as `o` in the printout.

   The labyrinth is to be interpreted as a graph, where all the possible locations are nodes and the possible routes are edges between nodes. Notice that it is not necessary to present the graph as adjecancy list or adjecancy matrix. Some other kind of representation may be more conventient way of implementing the graph and its edges.