CG2: Sheet 1

Exercise 2: Theory

a) Median of n numbers

It is possible to find the median of a set in linear time for the average case using the *quickselect* algorithm with $k = n/2$

0. Suppose we have a list of length n, and are looking for the k-th element
1. If the length of the list is 1, return the list element as the median.
2. Pick an index from the list at random. The element at this position is called the **pivot**
3. Create 2 lists, *lesser_piv* and *greater_piv*.

- Elements less than or equal to the pivot go in *lesser_piv*
- Elements strictly greater than the pivot go in *great_piv*

4. Distinguish between the following cases

- If **lesser_piv** contains more than or equal to k elements, go to line 1, with $list = lesser\_piv$ and $k = k$.
- if **greater_piv** contains more than k elements, go to line 1, with $list = greater\_piv$ and $k = k - len(lesser\_piv)$

This algorithm executes in $O(n)$ in the average case. The reason for this is that, assuming the average randomly chosen pivot halves the size of the list that is being searched each iteration, each recursion operates on a list takes half as long as the previous one. Searching the list takes linear time dependant on its size, thus the runtime is

$$O(n + \frac{1}{2}n + \frac{1}{4}n + \frac{1}{8}n + \dots) \equiv O(2n) \equiv O(n)$$

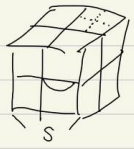
However, in the *extremely* unlucky worst case that the pivot point is chosen in a way that

$$len(lesser\_piv) = 1, len(greater\_piv) = n - 1$$

each iteration, the algorithm would take $O(n+n-1+n-2\dots) \equiv O(n(n+1)/2) \equiv O(n^2)$

b) Max. depth of Octree

#2.



each octant must have a side length at most ε, which is the smallest size and has the maximum depth. Each division reduces the size of the cell by 1/2. Therefore, let k the depth of octree,

$$\frac{S}{2^k} <= \varepsilon, \quad \frac{S}{\varepsilon} <= 2^k, \quad k => \log_2\left(\frac{S}{\varepsilon}\right).$$

$\therefore$ the maximal depth of octree is $\log_2\left(\frac{S}{\varepsilon}\right)$

c) K-nearest neighbor in linear runtime

In a worst-case scenario, the search performance of k-nearest neighbor search for all spatial data structures can lead to $O(n)$ runtime complexity. This is the case, when:

- *Grid* has highly uneven data distribution
- *Octree* is highly imbalanced
- *KD-tree* is highly imbalanced

For example all data points p_i are arranged in a straight line, where the trees collapse into a linked list. Searching for the point q at the end of the line will result in $O(n)$ complexity.

Group 6: Chanyoung Kim, Jiyun Park, Duc Thinh Nguyen, Richard Hanß