Beihang University
School of General Engineering

Beijing, Nov. 15th, 2019
Sebastian Wandelt, Pengfei Yi, Junyu Zhao

Computer Science and Programming
Lab Class 9

Task 1. k-selection - naive - first attempt based on sorting *(10 minutes)*

Implement a function *kselect_naive1* to select the k-th smallest element in an unsorted list which runs in $\Theta(n^2)$. The function receives a list L as input and returns the k-th smallest element in L. You can either use sorting or use your own algorithm.

Task 2. k-selection - naive - second attempt based on sorting *(10 minutes)*

Implement a function *kselect_naive2* to select the k-th smallest element in an unsorted list which runs in $\Theta(n * logn)$. The function receives a list L as input and returns the k-th smallest element in L.You can either use sorting or use your own algorithm.

Task 3. k-selection - random *(20 minutes)*

Implement a function to select the k-th smallest element in an unsorted list using the random selection strategy introduced in the lecture. If you forget about the concept, go back to the slides (Page 12, 13 in slides of Lecture 9) for a short review. Please follow these steps:

[1] Implement a function (several lines of code) that randomly selects an element in a list. The function should have one list as its input and the selected element's value as its output.

[2] Implement a function for partitioning. Following the code in [1], your function should compare all the elements in a list with the selected element (p) and put the elements larger than p and smaller than p in two separate lists.

[3] Based on the solutions for [1] and [2], implement a recursive function *kselect_random*. Please think about the cases for (left/right/pivot) and how to adapt the value of k without looking at the slides.

[4] Now you have implemented the random method for k-selection. With the analysis from the lecture, the time complexity of this method is in $O(n^2)$. Is this method faster or the naive method faster? Explain your answer and idea.

Task 4. k-selection - median of median *(remaining time)*

Implement a function *kselect_medianOfMedian*, which follows the idea presented in the last lecture, allowing to find the k-th smallest element in $\Theta(n)$.