Lab 6 - Selection

Collaboration level 0 (no restrictions). Open notes.

1. **Majority element:** Let A be a list of n (not necessarily distinct) integers. Describe an $O(n)$-algorithm to test whether any item occurs more than $\lceil n/2 \rceil$ times in A.

   (a) You may assume that the integers are in a small range, $K = O(n)$.

   (b) Come up with a general solution, without making any additional assumptions about the integers (in particular you may not assume that the range is small). Hint: use Select()

   (c) (challenge, optional) Come up with an algorithm that uses $O(1)$ additional space.

2. **k-smallest:** Let A be an array of n elements, assumed to be distinct. Develop an algorithm that computes the k smallest elements in A in $O(n + k \lg n)$ time.

   Note: we have seen solutions for this problem using sorting and using a heap; for this lab, come up with a solution that uses SELECT()*.*

3. **Select quantiles:** Given an unsorted sequence S of n elements, and an integer k, we want to find the $k-1$ elements that have rank $\lceil n/k \rceil$, $2\lceil n/k \rceil$, $3\lceil n/k \rceil$, and so on, up to $(k-1)\lceil n/k \rceil$.

   For example, if $k = 8$, we want to find the $n/8$th, $n/4$th, $3n/8$th, $n/2$th, $5n/8$th, $3n/4$th, and $7n/8$th smallest ellements.

   (a) Describe the "naive" algorithm that works by repeated selection, and analyze its running time function of n and k (do not assume k to be a constant).

   (b) Describe an improved algorithm that runs in $O(n \lg k)$ time. You may assume that k is a power of 2. After you describe it, argue why its running time is $O(n \lg k)$.

   We expect: high-level pseudocode and analysis