Order Statistics

CENG 112 - Data Structures

Ersin Çine
ersincine@iyte.edu.tr

Department of Computer Engineering
İzmir Institute of Technology

18.05.2017

Order Statistic

If our sample is $\{5, 3, 8\}$ then

- (0^{th}) smallest value is 3,
- 1^{st} smallest value is 5,
- 2^{nd} smallest value is 8.

k^{th} smallest value of a statistical sample
is called k^{th} **order statistic**.

An algorithm to find the k^{th} smallest value of an array
is called a **selection algorithm**.

k is called **rank** here (note that we use the zero-based numbering).

Offline Selection

Problem: We have an unordered array x of comparable values (e.g. numbers) of size n. What is the k^{th} smallest value in x?
e.g. What is the 3^{rd} smallest value in $\{27, 14, 19, 71, 8, 45, 11, 17, 69\}$?

- A naive approach: Sort the array in $\mathcal{O}(n \log n)$ and return $x[k]$.
- Quickselect: There is no need to sort the whole array. We need only arrange the values so that for all i $<$ k, x[i] $\leq$ x[k] and for all i $>$ k, x[i] $\geq$ x[k]. Then we return $x[k]$. This is done in $\mathcal{O}(n)$ in the average case using the partition function. Demo: `http://algs4.cs.princeton.edu/lectures/23DemoQuickSelect.pdf`

Online Selection

Problem: We have a stream of comparable values. At any time we want to know the k^{th} smallest value. e.g. Each second we generate a random number (we may not store them) and we want to see the 3^{rd} smallest number among the generated numbers whenever we look at the screen.

– Use a sorted array in increasing order of size $k+1$ to store the $k+1$ smallest values of the stream. The current k^{th} smallest value is the last value of this array (access time is $\mathcal{O}(1)$). For every new value, if the value is larger than the current k^{th} smallest value, ignore it. Otherwise, remove the current k^{th} smallest value from the array and insert the new value in sorted order. This is done in $\mathcal{O}(k)$.

– Use a max heap of size $k+1$ to store the $k+1$ smallest values of the stream. The current k^{th} smallest value is the root of the heap (access time is $\mathcal{O}(1)$). For every new value, if the value is larger than the current k^{th} smallest value, ignore it. Otherwise, replace the root with the new value and call downheap. This is done in $\mathcal{O}(log k)$.

Some Details

- Quickselect is linear (i.e. $\mathcal{O}(n)$) in the average case but quadratic (i.e. $\mathcal{O}(n^2)$) in the worst case. If the array is distributed uniformly at random, the worst scenario is very rare, therefore it will not be a problem. However, in many practical applications, the array is not distributed uniformly at random and this makes the worst case more likely. In such an application, use **randomized quickselect** (e.g. preshuffle or random pivot) instead.
- There is another offline selection algorithm called **median of medians** which also uses the partition function. It is linear in the worst case. But the constants are very high. Therefore it does not work well in many practical situations.

Exercises

- (*) Modify onlineselection.cc in order to find the k^{th} largest value (using a min heap).
- (**) Implement median of medians in offlineselection.cc
- (***) Design an efficient algorithm to find the median in a stream (Note that k is not fixed).