Randomized Algorithms (RA-MIRI): Assignment #2

1 Statement

In this programming assignment you will have to study experimentally the performance of a variant of quickselect called *sesquickselect*. Sesquickselect aims at reducing cache misses and thus improve running time of the standard quickselect algorithm. To this end, sesquickselect will randomly pick two pivots for each recursive stage, use one of them or both to partition the current subarray and continue recursively in the part that contains the sought element. Sesquickselect uses a parameter $\nu \in [0, 1/2]$ as follows. Let $\alpha = i/n$ if we are looking an element with current rank i in a subarray of size n (notice that i and n will be changing as we make recursive calls). Then if $\alpha < \nu$ we partition the array around the smallest of the two pivots and if $\alpha > 1 - \nu$ then we partition the array around the largest of the two pivots. In these two cases we compare the position of the pivot after partition with the position of the sought element and proceed recursively to the left or the right, as necessary. If $\nu \leq \alpha \leq 1 - \nu$ then the subarray is partitioned around the two pivots using Yaroslavskiy-Bentley-Bloch (YBB) dual-pivot partitioning, which will produce three parts, and we continue recursively into the one that contains the sought element.

Check the references below to learn more about the algorithm and about YBB partitioning (more specifically references [1] and [2]). To measure the performance of *sesquickselect* we shall use *scanned elements*, which give a very good approximation to the number of cache misses (after dividing by the cache size) that the algorithm incurs. In the classical partitioning algorithm "scanned elements = comparisons = $n - 1$", but this is not the case with YBB partitioning. In YBB partitioning the number of scanned elements equals "$n - 2 +$ size of the leftmost part". A detailed explanation of YBB partitioning can be found in Figure 1 of [1].

In your experiments you should produce a big number T of large arrays, e.g., with $n = 30000$ elements, each one containing a random permutation. Apply sesquickselect to the arrays with various ranks $i \in [1..10000]$, e.g., for $i = 1, 100, 200, \ldots, 29900, 30000$ and obtain the average scanned elements

$$\hat{S}_n^{(i)} := \frac{1}{T} \sum_{r=1}^{T} \hat{S}_n^{(i)}(r),$$

where $\hat{S}_n^{(i)}(r)$ denotes the number of scanned elements to select the i-th element in the r-th array of size n. Let $i_x = \lfloor x \cdot n \rfloor$; plot $\hat{S}_n^{(i_x)}/n$ as a function of $x \in (0,1)$ and compare with the theoretical prediction

$$f(x) = \lim_{n\to\infty} \frac{S_n^{(\lfloor xn \rfloor)}}{n}, \qquad x \in [0,1].$$

Here $S_n^{(i)}$ is the theoretical value of the expected number of scanned elements when selecting the i-th smallest elements out of n. The values $\hat{S}_n^{(i)}$ that you obtain through experiment are estimates of the true values $S_n^{(i)}$. You should be getting plots similar to those Figure 14 in [3] (experimental results), and Figures 2 and 3 of [3] or Figure 2 (right) of [2] (theoretical predictions).

Setting $\nu = 0$ you get quickselect with pure dual-pivot partitioning (you use two pivots and YBB partitioning at every stage), whereas setting $\nu = 1/2$ you will always partition using only one pivot—this variant is known as *proportion-from-2* as the pivot that is used comes from a sample of 2 elements on each stage (see [3]). Besides the special values $\nu = 0$ and $\nu = 1/2$, you should do experiments for other values of ν, and in particular, try to check experimentally (very approximately) that the optimal value of ν is $\nu^* \approx 0.265\ldots$. Reference [2] gives precise formulas for the function $f(x)$ for sesquickselect; the function depends on ν albeit we don't make this dependence explicit, and unless $\nu = 0$ or $\nu = 1/2$, the function has three pieces:

$$f(x) = \begin{cases} f_1(x) & \text{if } x < \nu, \\ f_2(x) & \text{if } \nu \le x \le 1-\nu, \\ f_1(1-x) & \text{if } x > 1-\nu. \end{cases}$$

The values of $f_1(x)$ and $f_2(x)$ are given in Section 7 of [2], and the necessary constants in Appendix D. If you want to plot the theoretical curves $f(x)$ (which also depend on ν) prepare some small program (perhaps in Mathematica, Maple or a similar computer algebra system) that generates the plots or to produce files with $(x, f(x))$ pairs (one file for every ν) to be used later by your plotting system or package. Plotting the theoretical curves and the estimation that you get from the experiments alongside might be useful (but avoid plotting too many curves in one single plot!). It can also be instructive to plot the errors $|f(x) - \hat{S}_n^{(i_x)}/n|$, and to estimate the variability of the measurements, i.e., compute the sample variance

$$\hat{V}_n^{(i)} = \frac{1}{T-1} \sum_{r=1}^{T} (\hat{S}_n^{(i)}(r) - \hat{S}_n^{(i)})^2$$

which is an unbiased estimator of the variance of the number of scanned elements to select the i-th out of n. The square root of $\hat{V}_n^{(i)}$ is not an unbiased estimator of the standard deviation (because $\mathbb{E}\left\{\sqrt{X}\right\} \neq \sqrt{\mathbb{E}\{X\}}$!), but is not far away, especially if T is large. Plotting a bar of length $\sqrt{\hat{V}_n^{(i)}}/n$ (which is a function

of i) centered around the corresponding average value $\hat{S}_n^{(i)}/n$ gives a good idea of the amount of variability in the measurements.

1. Sebastian Wild. *Dual-pivot and beyond: The potential of multiway partitioning in quicksort*. Information Technology 60(3): 173–177, 2018. Available from: `https://www.wild-inter.net/publications/wild-2018b.pdf`

2. Conrado Martínez, Markus Nebel and Sebastian Wild. *Sesquickselect: One and a half pivots for cache-efficient selection*. In Marni Mishna, J. Ian Munro, editors, Proc. of the 16th Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2019, San Diego, CA, USA, January 6, 2019. Pages 54–66, SIAM, 2019. Available from: `https://epubs.siam.org/doi/10.1137/1.9781611975505.6`. On-line full version available from `https://arxiv.org/abs/1810.12322`

3. Conrado Martínez, Daniel Panario and Alfredo Viola. *Adaptive sampling strategies for quickselect*. ACM Transactions on Algorithms 6(3), pages 53:1–53:45, 2910. Available from: `https://dl.acm.org/doi/10.1145/1798596.1798606`

N.B. These papers can be accessed from the UPC library website (`https://bibliotecnica.upc.edu/ebib`) identifying yourself as UPC student; if you experience any problems to access them or you need access to some related material contact me (DM in Slack or email to `conrado@cs.upc.edu`).

2 Instructions to deliver your work

Submit your work using the FIB-Racó. The deadline for submission is November 12th, 2023 at 23:59. It must consist of a zip or tar file containing all your source code, auxiliary files and your report in PDF format. Include a README file that briefly describes the contents of the zip/tar file and gives instructions on how to produce the executable program(s) used and how to reproduce the experiments. The PDF file with your report must be called `YourLastName_YourFirstName-2.pdf`,

N.B. I encourage you to use LaTeX to prepare your report. For the plots you can use any of the multiple packages that LaTeX has (in particular, the bundle TikZ+PGF) or use independent software such as matplotlib and then include the images/PDF plots thus generated into your document.