

## CS 201, Spring 2021

### Homework 4

**DUE: May 13, Friday @23:55**

**Description:** In this assignment, you will write a C++ program that **finds the  $k^{\text{th}}$  largest number** among a set of  $N$  numbers. However, this time, you will use a **variation of the quicksort** algorithm to implement the program. The details can be found below.

#### 1) The Input and Expected Output

The input and the expected output of the program is the same as described in Homework 1 and 3. The program will take the **type of algorithm** to be applied,  $k$  (a number less than or equal to  $N$ ). Then it will take  $N$  followed by a list of  $N$  **numbers**. As output, it will print out the  **$k^{\text{th}}$  largest number** and the **total elapsed time** for the completion of the algorithm.

As the only difference from Homework 1 and 3, **the algorithm type** can be 1, 2, 3 or 4. You can modify and reuse the test generator program to generate test inputs. You can also just modify the sample test inputs for Homework 1 to test the same set of numbers with algorithm type being 4.

Your code will only be tested with algorithm type 4.

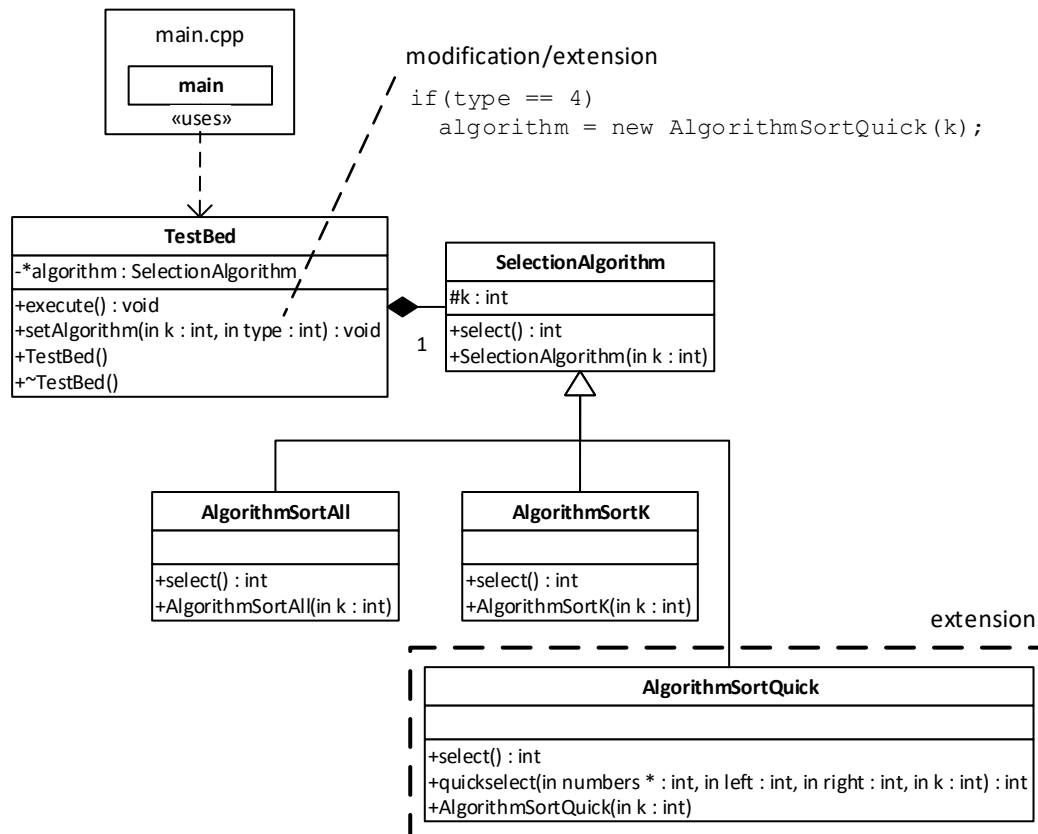
#### 2) The Algorithm: quickselect

You will implement a variation of the **quicksort** algorithm to find the  $k^{\text{th}}$  largest number among a set of numbers  $S$  in  $O(N)$  time on average, where  $N$  is the total amount of numbers, i.e.,  $|S|$ . The outline of the algorithm, **quickselect** is as follows.

- If  $N \leq 10$ , sort all the numbers (using insertion sort) and return the  $k^{\text{th}}$  number.
- Else, pick a pivot number,  $v \in S$ .
  - Partition  $S - \{v\}$  into  $S_1$  and  $S_2$ , as done in quicksort.
  - If  $k \leq |S_1|$ , then the  $k^{\text{th}}$  largest number must be in  $S_1$ . Return  $\text{quickselect}(S_1, k)$ .
  - If  $k = 1 + |S_1|$ , then the pivot,  $v$  is the  $k^{\text{th}}$  largest number. Return  $v$ .
  - Otherwise, the  $k^{\text{th}}$  largest number lies in  $|S_2|$ , and it is the  $(k - |S_1| - 1)^{\text{st}}$  largest number in  $S_2$ . Return  $\text{quickselect}(S_2, k - |S_1| - 1)$ .

### 3) The (Extended) Design

You can reuse the design of Homework 1 and/or Homework 3. You just need to add one extra class, *AlgorithmSortQuick*, which extends from the *SelectionAlgorithm* class and overwrites the *select* method to implement the new algorithm. The *select* method within the *AlgorithmSortQuick* class should make a call to the recursive *quickselect* method. It takes an array of numbers, the left and the right index of the partition of interest, and *k*. The overall design is depicted below.



You might need to make further small modifications in function *main* and the *TestBed* class to accept 4 as the algorithm type. This is because, previously the algorithm type was assumed to be 1, 2 or 3. For instance, you should make a modification/extension within the *setAlgorithm* method of the *TestBed* class. If the *type* argument is 4, an object of type *AlgorithmSortQuick* should be assigned to the *algorithm* member variable. The rest of the design and implementation can be reused as is.

If you have not submitted Homework 1 or Homework 3, then you have to implement at least the *main* method, the *TestBed* class and the *SelectionAlgorithm* class as described in the assignment description of Homework 1.

#### 4) Submission

You will submit this homework via the LMS system. You should follow the file-naming conventions and guidelines below.

- You should submit your source files as a **ZIP** archive file (**NOT** RAR or other formats). The name of the file should be in format “<USER-ID>\_hw<HOMEWORK-NR>.zip”. For example, if your username is vy1043, then the name of the submitted file should be “vy1043\_hw4.zip”. Pay attention that all the letters are in lower-case. ZIP archive is supposed to contain **just the source files**, no folders are allowed by any means.
- The contents of the ZIP file should be as follows:
  - **main.cpp** (includes the *main* function)
  - **TestBed.h** (TestBed class definition)
  - **TestBed.cpp** (TestBed class implementation)
  - **SelectionAlgorithm.h** (SelectionAlgorithm class definition)
  - **SelectionAlgorithm.cpp** (SelectionAlgorithm class implementation)
  - **AlgorithmSortQuick.h** (AlgorithmSortQuick class definition)
  - **AlgorithmSortQuick.cpp** (AlgorithmSortQuick class implementation)
  - **AlgorithmSortK.h** (AlgorithmSortK class definition)
  - **AlgorithmSortK.cpp** (AlgorithmSortK class implementation)
  - **AlgorithmSortAll.h** (AlgorithmSortAll class definition)
  - **AlgorithmSortAll.cpp** (AlgorithmSortAll class implementation)
  - **AlgorithmSortHeap.h** (AlgorithmSortHeap class definition)
  - **AlgorithmSortHeap.cpp** (AlgorithmSortHeap class implementation)
  - **BinaryHeap.h** (BinaryHeap class definition)
  - **BinaryHeap.cpp** (BinaryHeap class implementation)
- Late submissions and C++ files that do not compile are **not** accepted.
- You can resubmit your homework (until the deadline) if you need to.
- Make sure that your program does **not** include commands specific to a development environment, e.g., `system("pause")` or `#pragma once` in Visual Studio.

\* **Optional:** The last 8 files are to be submitted if implemented as an extension of Homework 1 and/or Homework 3. The corresponding algorithms will not be tested for evaluating Homework 4. The algorithm type will always be set as 4 in the test cases.