SAKSHI SHAH (001003644)

**Program Structures & Algorithms**

**Spring 2021**

* **Task :**

Implement Floyd’s trick in the *PriorityQueue* class (in our repository).

You will create a new method “snake” (as in snakes and ladders) for the fast version of *sink*. The textbook has a very short explanation of Floyd’s trick, and we have covered it in class, but if you need more understanding, let me know.

So, you will update the *take* method to use *snake*+*swim* instead of just *sink*. And of course you’ll need to pass a *Boolean* into the constructor to set whether Floyd is on or off (default off).

Thoroughly test this via unit tests. Snake will be private so you will have to use the *PrivateMethodTester* .

Once tested, I want you to run benchmarks. Add appropriate entries in *config.ini* so that you can turn Floyd on or off.

Then, of course, run benchmarks with Floyd on/off and compare.

You should do this for two different key types: int and String, where the Strings are English words.

* **Output:**

1. I have implemented a private method (Snake) which sinks the key present at the root, all the way to the bottom. This takes one comparison per row. I have then used Snake + swimUp in the take method instead of sink. Upon doing so, I have all my unit test running. I have also written new unit tests (testTake4, testTake5, testTake6), which have different numbers of elements in the priority queue. *These unit tests are also running successfully.*
2. Next, I have implemented Boolean Floyd in the constructor of the **PriorityQueue.java**. It takes values from **config.ini** which is present in */src/test/resources/config.ini*. We can turn Floyd true/false from the config file. *This is also working properly and as expected.*
3. If we have a priority queue with 4 rows(i.e., 15 elements), the number of compares used with Floyd's false is 6 to bring the element to the last row. We can see that the number of compares is reduced to 4 if only 4 rows are present and we use Floyd's Trick(Snake + swimUp) (Row 2- 1 compare, Row 3- 1 compare, Row 4- 1 compare, and for swimUp 1 compare to check if a parent is greater than the key, and this is true so no more compares).
4. For **Benchmarking**, I have created a file **PriorityQueueBenchmark.java**which is present in */src/main/java/edu/neu/coe/info6205/util/PriorityQueueBenchmark.java.*I have benchmarked the number of compares used to delete all the elements present in the Priority Queue with Floyd's true/false I have tried this with different numbers of words and Integers. I deleted all the items in the priority queue and ran it for *nRuns* times, and later printed the average result. I did it with *4000,8000,16000 and 32000*different types of Strings and Integers.

* **Unit tests result:**
* I have attached the image of benchmarking with Floyd's true and false and the result of the unit test cases with Floyd's true and false.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, email

Description automatically generatedGraphical user interface, text, application

Description automatically generated