**Report (SP6)**

**Group : g39 Authors: Khushbu Patil**

**Vatsal Patel**

**Shruti Shetye**

Q8. Compare the running times of the following two algorithms for the problem of finding the k largest elements of a stream:

(a) Use Java's priority queue to keep track of the k largest elements seen

(b) Use your priority queue implementation (problem 5) using the replace()

operation in that implementation, instead of delete+add to update PQ.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Java's Priority Queue** | | **Custom Priority Queue** | |
| **n** | **k** | **Increasing ordered (msec)** | **Randomly ordered**  **(msec)** | **Increasing ordered (msec)** | **Randomly ordered (msec)** |
| 1M | 2000 | 432 | 147 | 452 | 93 |
| 5M | 2000 | 1504 | 499 | 1947 | 308 |
| 10M | 2000 | 3271 | 849 | 3705 | 680 |
| 25M | 2000 | 5482 | 1949 | 9965 | 1988 |
|  | | | | | |
| 1M | 3000 | 445 | 147 | 494 | 108 |
| 5M | 3000 | 1415 | 381 | 2012 | 339 |
| 10M | 3000 | 3552 | 907 | 3974 | 682 |
| 25M | 3000 | 8029 | 2015 | 10312 | 1864 |

* It is clear from the results in table above, custom implementation of priority queue performs better on randomly ordered arrays while java’s own implementation of priority queue performs better on increasing ordered arrays.