**GROUP NUMBER:** 97

**GROUP MEMBERS:** Umberto Bianchin, Massimo Vettorello

**AVAILABLE INPUTS:** Input files are available in the hdfs file system: /data/BDC2425/artificial1M7D100K.txt and /data/BDC2425/artificial4M7D100K.txt

**PART 1:** The goal of this test is to assess the scalability of the standard and fair implementations. The test must be performed on file artificial4M7D100K.txt. However, if your implementation is slow (i.e., taking more than 10 minutes for the slowest run), you can use the smaller file artificial1M7D100K.txt. You must use the following parameters: L=16, K=100, M=10.

Fill in the following table.

**Name of used file:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SCALABILITY WITH RESPECT TO NUMBER OF EXECUTORS** | | | |
| **Number of executors** | **Spark Lloyd’s implementation** | **MRFairLloyd** | **MRComputeFairObjective** |
| 2 |  |  |  |
| 4 |  |  |  |
| 8 |  |  |  |
| 16 |  |  |  |

**General hints:**

* Remember that Spark uses the lazy evaluation for constructing an RDD. Therefore, be sure to include an action on the final RDD when you take running times.
* Any used RDD in your program should be cached.
* Do not include the reading of the input in your running times.

**PART 2:** Describe the program GxxGEN.java or GxxGEN.py that you have implemented for point 5 of the specifications. Include a brief high-level description of your program and the constraints, if any, on the input parameters (e.g., the minimum number of points *N*).