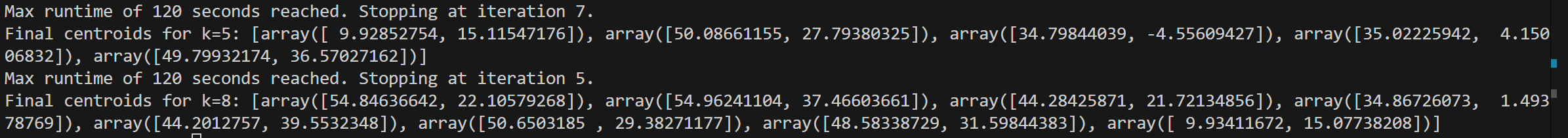
Q1 & Q2. The Java and Python are used for cross-validation of the results. The code is available in the compressed file. For Java Docker was used to provide Hadoop environment.

The Java version console output is provided as a txt file named dockerConsoleLog.txt, which contains the result of the task 1 and task 2 as well. For task 2, since Docker is used, to build and run the docker image / container, refer to the README.md file.

For Python, the task 1 result is provided as **line\_count\_output.txt**. As for task 2, below is a screenshot of task2 result from the console:



Q3. **Advantages:** MapReduce could speed up the convergence of K-Means since it distributes the workload.

**Disadvantages:** When the dimensions comes up, it could be struggling with convergence and communication would be a problem as well. Also, Hadoop is complex to set up in a local environment unless EMR or other similar technology is used

Q4. **Yes**, because it reduces distance comparisons by creating overlapping subsets (namely canopies) and only comparing points within the same canopy. In terms of high dimensions, we could use an approximate measurement for efficiency.

Q5. **Yes**. In the Map phase, each data point would be assigned to canopies based on the efficient distance metric. In the Reduce phase, data points within each canopy would be sub-clustered with a more accurate distance metric.

Q6. **Yes**. First, use MapReduce to perform Canopy Selection as described above. Then, in another MapReduce job to K-Means clustering could be used on the points within each canopy, using a more accurate distance metric.