Grading (8 points)

In this assignment, you are required to complete the following:

For Part 2, please fill the table below based on the execution results of your Word Count program (1

Word	for	called	to	the
Occurrence	7	4	29	46

Upload the code and execution log in Part 2. (1 point)

I PUT THEM ON OTHER FILES IN THIS FOLDER

For Part 3, please fill the table below based on the execution results of your PageRank program. (2

Node	448126	49717	375375	63087
Rank	11.83	4.38	50.91	0.64

For Part 3, Task 2, explain the custom RDD partitioning you implemented and the changes it brings

I chose to use the repartition method to partition the DataFrame instead of partitionBy(). Specifically,

I used two strategies:

- Fixed Number of Partitions (repartition(4)):

```
edges_df = edges_df.repartition(4)
out_degree_df = out_degree_df.repartition(4)
ranks_df = ranks_df.repartition(4)
```

- Partitioning by Column (repartition("from")):

```
edges_df = edges_df.repartition("from")
out_degree_df = out_degree_df.repartition("from")
ranks_df = ranks_df.repartition("id")
```

Reasons for Choosing repartition(4) and repartition("from"):

The Spark cluster consists of two EC2 instances (vm1 and vm2), each with 2 vCPU cores, totaling 4 cores. By repartitioning into 4 partitions, each core can process one partition simultaneously, maximizing resource utilization.

Partitioning based on the 'from' column ensures records with the same 'from' value are in the same partition. This minimizes data shuffling during join operations in the PageRank algorithm.

- repartition(4): ensures that all processing units are utilized efficiently.
- repartition("from"): Ensures related data resides in the same partition for efficient join operations.

For Part 3 Task 3, explain the changes after the Worker process is killed. (1 point)

If I neither clear memory nor terminate workers, jobs initially start with one parallel task, gradually increasing until reaching stable parallelism with minimal skipped stages. Eventually, all tasks complete successfully.

When I clear worker 2's memory midway, all parallel jobs momentarily disappear. Some resume, while others restart as new jobs. The skipped stages and shuffle operations on disk increase significantly, leading to memory exhaustion and some task failures.

Upon terminating a worker, an executor is removed. Immediately, failed tasks and stages appear, causing delays in restarting jobs. Parallelism decreases as tasks queue on the remaining executor, overloading it and sharply increasing task failures.

Upload the program and the corresponding execution logs for Task 1 and 2 in Part 3. (2 points)

I PUT THEM OTHER PLACE IN THIS FOLDER