Task 1, which implemented the basic PageSort algorithm with no custom partitioning, completed in 8.4 minutes. Notably, a good chunk of this runtime (specifically 2.8 minutes worth) is taken up not by the PageSort iterative algorithm, but by the initialization of linksRDD and ranksRDD. This makes sense, as these stages contained the largest amount of data partitioning, from an input of 105 MiB to a Shuffle Write of ~ 40 MiB. In order to improve performance through repeated calls, linksRDD and ranksRDD were stored to cache and persistence respectively. In contrast to the setup stages, the iterative stages (calculating page contribution and summing those contributions to update ranks) ran much faster, completing in 10-15 seconds (in other words, about 25 seconds per iteration). However, because of the lack of custom partitioning, each iteration brought an increase to the number of partitions, which in turn increased the number of tasks in each stage (and Shuffle Read/Write sizes). As such, the last few iterations took longer (around 50 seconds) to complete.

In Task 2, I added the partitionBy() function to the initializations of linksRDD and ranksRDD, splitting the data into 8 even chunks. This slightly increased the runtime of the pre-iteration stages to 3.7 minutes. However, this also had the effect of vastly improving the distributional efficiency of the job, with Task 2 completing in 15 stages as opposed to Task 1's 23. Additionally, each iterative stage only had 8 tasks, so the total number of tasks was much lower as well (119 compared to 332), which in turn meant the iteration stages completed much faster (consistently ~ 20 seconds). All of this contributed to an overall faster time of 6.9 minutes.

Task 3 was identical to Task 1 in terms of code, but during execution I killed the vm2 Worker node at around stage 12. This node failure, occurring during the iteration, meant that stages 2-12 had to be rerun (with half the working nodes no less); three minutes of work were essentially gone. Fortunately, due to the caching and persisting of linksRDD and ranksRDD, these retries were able to complete in faster time, only taking about 20 seconds per iteration. On the flip side, once the execution had moved stages not previously done, the lack of a second executor caused the runtime of each iteration to significantly increase to about 40 seconds (60 seconds in the last couple iterations). Taken together, the lost time of failed tasks and worsened performance caused Task 3 to have an overall slower time of 12 minutes.