|  |  |  |  |
| --- | --- | --- | --- |
| Size | Nanoseconds Computing (ns) | Milliseconds Computing (ms) | Nanoseconds  Finding (ns) |
| 51 | 3131636000 | 3131 | 62000 |
| 251 | 3136688000 | 3136 | 40000 |
| 551 | 3094462000 | 3094 | 54000 |
| 1051 | 3141766000 | 3141 | 51000 |
| 10051 | 3083725000 | 3083 | 54000 |
| 100051 | 3107297000 | 3107 | 55000 |
| 1000051 | 3438722000 | 3438 | 61000 |
| 10000051 | 6702140000 | 6702 | 51000 |

Time comparisons among different sizes for HashMap

Note: Computing Means ComputingAdjacentWords, Finding means FindingAdjacentWords. Word used to find was bead, so finding is based on a successful find.

Note2: This was compiled on Cygwin, and unnoticed until submission for this assignment (When I tested on Linux an hour before it was due), I realized that the computing and finding speed was actually a bit faster on Linux (Around 1700000000 ns or 1800000000 ns). However the pattern was similar, not changing much until 10000000 size, where it suddenly doubled, so I decided not to change it since I lacked the time to anyhow, and hope this isn’t an inconvenience since the computation pattern is still the same.

The sizes, based on my calculations, did not seem to change anything until a size of 1000000 was tested, which slightly increased the amount of time Computing the adjacent words, but still didn’t change much in regards to Finding. 10000000 doubled the Computing time, a more significant change, but still didn’t effect finding bead. The size at this point may of made hashing counterproductive, or effecting the Quadratic Probing as a whole, showing that too large a size does not serve as a pure positive for Quadratic Probing despite it performing better with large sizes.