|  |  |  |  |
| --- | --- | --- | --- |
|  |  | input time |  |
|  | Small | Medium | Large |
| ArrayList | 979.2 | 6135.2 | 73533.4 |
| BST | 493 | 2192 | 14987.7 |
| Hash | 433.8 | 2046.6 | 15143.4 |

We expected the array list to dominate both the binary search tree and hash map for all file sizes because of its average time for adding to the array being linear, while trees are n\*logn time and hash maps use logn time on average. We also expected the search tree to dominate the hash map on larger files. Our expectations were correct, though the domination ends when we examine the large file size. Below are our observations for output. We thought the outputs would show a similar decline in output time to what was demonstrated in recording the input. However, after 5 runs of every file with output of one thousand, ten thousand, and twenty thousand words we came to discover that the difference in output time is very small and even trivial in all cases except when printing 20,000 words from the King James Bible; where the hash map shaved about 3500 msec (3.5 seconds) off the output time of both the array list and binary search tree.

Output: 1,000 10,000 20,000

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Small | | 89.2 | | 375 | | 903.8 | |
| ArrayList | | Medium | | 180.2 | | 1484.2 | | 3074.8 | |
|  | | Large | | 2188.2 | | 19259.2 | | 38292 | |
|  |  | |  | |  | |  | |
|  | Small | | 111.8 | | 411.4 | | 954.6 | |
| BST | Medium | | 219 | | 1507.4 | | 3075.8 | |
|  | Large | | 2199.8 | | 18995.2 | | 38158 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | small | 58.2 | 395.6 | 918.8 |
| Hash | medium | 208.6 | 1484.8 | 2996.6 |
|  | large | 2198.8 | 19067.4 | 34470.8 |