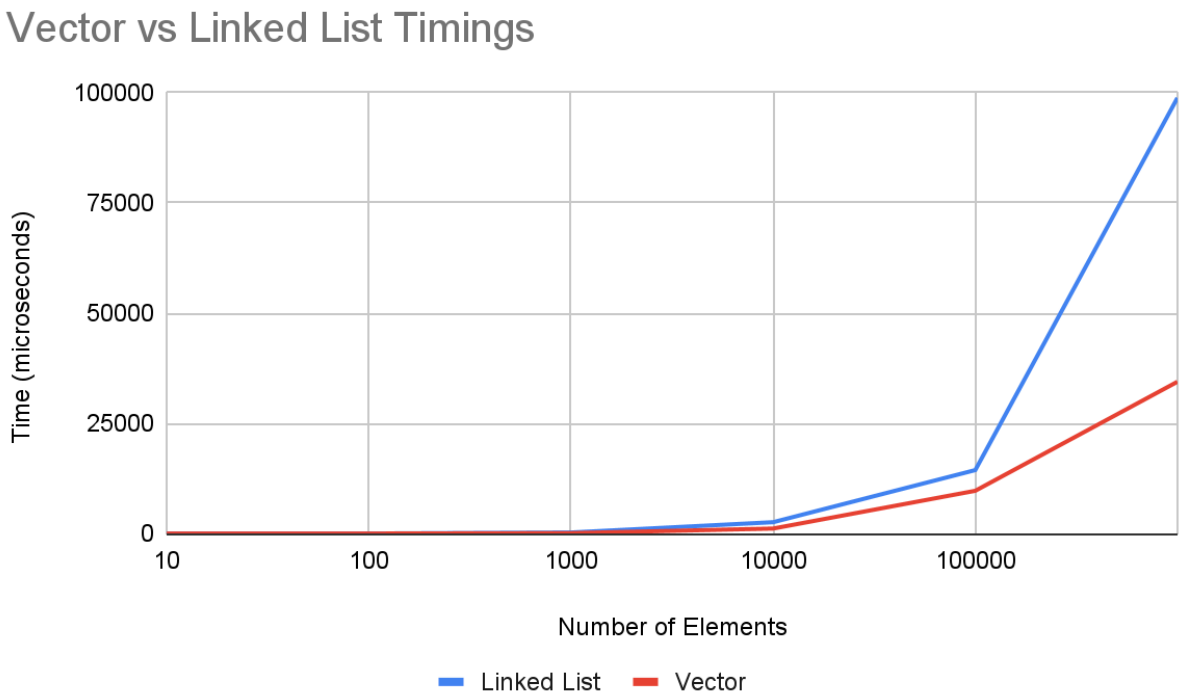
Cody Wu
133001517

LE3: Arrays vs Linked List



During the push operation, the vector has a theoretical time complexity of O(n) in its worst case and O(1) in its average case. On the other hand, the linked list has a theoretical time complexity of O(1) in all cases. In theory, the linked list boasts a better time complexity. However, it can be seen in the graph above that the linked list takes significantly more time to push compared to the vector.

This gap is due to the principle of spatial locality. The principle of spatial locality states that when memory is accessed, the nearby memory is also grabbed. This works in favor of the vector compared to the linked list. Vectors and arrays have dedicated memory that is kept together. This means that when something is accessed in the vector, the principle of spatial locality will also grab nearby memory. This allows for a much faster push time. The linked list uses memory in a more spread out way. This is better for conserving space and memory. However, it does not help its time complexity. Since not all of the data within a linked list is necessarily nearby, the memory has to be looked for which causes it to take a lot more time compared to the vector.

This conclusion can be observed when running Cachegrind. When running the vector program and recording its cache misses, it can be seen on Cachegrind that there are 140,045 L3 cache misses. When running Cachegrind on the linked list program, it can be seen that there are 1,010,606 L3 cache misses. There are significantly more cache misses on the linked list program.

The linked list program has to make a lot more requests to main memory which slows down its push operations.