3. Array vs. vector

We would use an array when we know the exact number of elements we will have. This avoids overhead with the vector’s dynamic growing as well as some other more minor features such as time and space needed for iterators and push_back markers.

4. Vector Performance

Vectors grow to have space for more data all the while keeping reference to the next unused data element at the end of the vector/array. This is where the push_back function places its data. This is an O(1) operation in the worst case; however, this does not take into account the situation where a reallocation happens due to a need to increase vector capacity. This in itself is considered an O(1) operation thus not changing the original conclusion of O(1) time complexity for a push_back. This is the case for an insert as well for when a reallocation of memory occurs.

An insert in its worst case is O(n) where n is the number of elements currently in the array/vector. Due to the fact that an insert allows data to be placed anywhere within the array everything at and after the desired array index will have to be shifted by one index to accommodate the inserted element. So, if we insert at the beginning of the array we will have to shift the entire array by one index.