Rachel Friedman
CISC 3412 | Programming Paradigms with C++ | Professor Xiang
April 8, 2021
Homework 3, Part 1

Evaluating performance of STL containers (unsorted)

Operation 1: End Insertion

In order of speed (from fastest to slowest):

- vector
- list
- unordered_set
- set

For end insertion, vector was the fastest and set was the slowest. The reason why vector was the fastest for this operation is because the capacity of a vector is initially double its size, so the memory (which is contiguous) for the new element has already been allocated. When adding to the end of a linked list, in addition to allocating memory for the element, the relevant pointers to the next and previous element must be reassigned/initialized. The reason why it's slowest when adding to a set is because a set is typically implemented as a binary search tree and stores its elements by following a specific order. In an unordered_set, the elements are not sorted in any particular order, but organized into buckets depending on their hash values. Therefore, insertion into an unordered_set is faster than insertion into a set which is ordered.

Operation 2: Beginning Insertion

In order of speed (from fastest to slowest):

- list
- unordered_set
- set
- vector

For beginning insertion, list was the fastest and vector was the worst by far. In order to add elements to the beginning of a vector, each of the existing elements must be shifted to the next position, which therefore causes the insertion process to be the slowest of each of the four containers. As mentioned for the previous insertion operation, insertion into an unordered_set is faster than insertion into a set which is ordered. Beginning insertion is the fastest for a list, because all that is required is memory allocation for the new element and the reassignment/initialization of the relevant previous and next pointers for the current head and the new head.

Operation 3: Find 10K Numbers

In order of speed (from fastest to slowest):

- unordered_set
- set
- vector
- list

For finding numbers, unordered_set was the fastest, and list was the slowest. An unordered_set is fastest because its elements are retrieved based on their hash values. A set is faster than a vector or list because its elements are stored as a binary search tree. For both a vector and a list, each of the elements must be searched until the correct one is found, and so that results in the slowest speed. Although time complexity is O(n) for both a list and a vector, searching a vector is slightly more efficient than a list because its elements are stored in contiguous memory so the next record is already in the cache, and the pointer can simply be easily advanced, while in a list the next pointer must be fetched.

Key takeaways

- Vector is the slowest (by orders of magnitude) for beginning insertions.
- If your program requires mostly insertion operations (where position matters, i.e. elements can't simply be added to the end), and very few lookup operations, use a list (or unordered_set) and definitely avoid vectors.
- If your program requires multiple lookups, avoid lists and vectors, and use an unordered_set. A set is second best, with an average time complexity of O(log n) compared to the O(1) time complexity of an unordered_set.
- For a set, time complexity for insertion is roughly equivalent for both beginning and end.
- For an unordered_set, time complexity for insertion is roughly equivalent for both beginning and end.
- A list and an unordered_set have a roughly equivalent time complexity for insertions, but an unordered_set is order of magnitudes faster when used for lookup.

Below are the execution times of these operations on my computer, as well as the average time complexities for these 4 containers.

Execution time (ms)	**End Insertion**	**Beginning Insertion**	**Find 10K Numbers**
`std::vector`	**7,000**	**2,250,068**	3,164,803
`std::list`	13,961	**20,155**	**4,644,959**
`std::set`	**74,800**	88,575	2,992
`std::unordered_set`	39,894	32,153	**998**

Key: **Best**

Worst

Avg Time Complexity	**End Insertion**	**Beginning Insertion**	**Find 10K Numbers**
`std::vector`	O(1)	O(n)	O(n)
`std::list`	O(1)	O(1)	O(n)
`std::set`	O(log n)	O(log n)	O(log n)
`std::unordered_set`	O(1)	O(1)	O(1)

Some sample outputs:

```
Execution time for inserting at end:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------7e+003-------------100.00%
          List---------------13961-------------199.96%
           Set---------------74800------------1071.33%
 Unordered Set---------------39894-------------571.38%

Execution time for inserting at beginning:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------2250068-------------100.00%
          List---------------20155---------------0.90%
           Set---------------88575---------------3.94%
 Unordered Set---------------32153---------------1.43%

Execution time for finding 10,000 numbers:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------3164803-------------100.00%
          List-------------4644959-------------146.77%
           Set----------------2992---------------0.09%
 Unordered Set-----------------998---------------0.03%
```

```
Execution time for inserting at end:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------9e+003-------------100.00%
          List---------------14962-------------166.69%
           Set---------------80783-------------899.99%
 Unordered Set---------------39893-------------444.44%

Execution time for inserting at beginning:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------2127425-------------100.00%
          List---------------10970---------------0.52%
           Set---------------78826---------------3.71%
 Unordered Set---------------32876---------------1.55%

Execution time for finding 10,000 numbers:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------2997822-------------100.00%
          List-------------4238235-------------141.38%
           Set----------------2013---------------0.07%
 Unordered Set-------------------0---------------0.00%
```

```
Execution time for inserting at end:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------8e+003-------------100.00%
          List---------------14779-------------193.92%
           Set--------------104055------------1365.37%
 Unordered Set---------------41917-------------550.02%

Execution time for inserting at beginning:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------2209445-------------100.00%
          List---------------10662---------------0.48%
           Set---------------85158---------------3.85%
 Unordered Set---------------31620---------------1.43%

Execution time for finding 10,000 numbers:
     CONTAINER              TIME (ms)           PERCENTAGE
        Vector-------------2592631-------------100.00%
          List-------------4823027-------------186.03%
           Set----------------3535---------------0.14%
 Unordered Set----------------1619---------------0.06%
```