Kovalenko Nikita 253

CODE ON GITHUB

PDF Report

1. Averaged Results Table

It shows the average time in milliseconds for each test

Operation	**Container**	**N = 8192**	**N = 32768**	**N = 262144**	**N = 500000**
PushBack	Vector	0.0135	0.0145	0.0926	0.1687
	Deque	0.0178	0.0171	0.1374	0.2839
	List	0.1590	0.1670	1.4752	2.7809
PushFront	Vector	2.3797	19.0868	1965.3611	7274.7058
	Deque	0.0228	0.0522	0.4444	0.8302
	List	0.0731	0.1687	1.4897	2.7444
RandomInsert	Vector	1.0749	10.0255	963.0080	3613.8538
	Deque	0.8463	6.4563	544.7531	2035.1665
	List	43.7512	1415.7750	129141.9445	556519.2675
RandomErase	Vector	0.6530	9.8737	954.5603	3556.0437
	Deque	0.5675	6.1176	478.6195	1824.5294
	List	21.5980	401.6145	18055.3699	93089.1048
IterateSum	Vector	0.0072	0.0291	0.2360	0.4443
	Deque	0.0021	0.0087	0.0675	0.1345
	List	0.0139	0.0362	0.3100	0.5897
RandomAccess	Vector	0.0019	0.0075	0.1298	0.2587
	Deque	0.0023	0.0122	0.1293	0.2602

Operation	Container	N = 8192	N = 32768	N = 262144	N = 500000
	List	37.4951	657.2533	37661.6632	140543.4162

To get the program to finish, the really slow tests were only run 1 time for the two biggest sizes, while the fast tests were run 10 times

---

2. Conclusions for Each Operation

- **PushBack: Adding to the end**
  - **Observation:** All three are pretty fast. But vector and deque are way faster than list.
  - **Conclusion:** vector and deque grab extra memory ahead of time, so adding is usually instant. list has to make a new box for every single element, and that extra work adds up.
- **PushFront: Adding to the front**
  - **Observation:** deque and list are super fast, almost instant. vector is terrible and gets crazy slow as the number of elements N grows.
  - **Conclusion:** This is the most important test. list and deque just change a pointer to say this is the new front. But vector has to physically move every single element one spot to the right to make room. This is a huge, slow job that gets worse and worse.
- **RandomInsert and RandomErase: Changing the middle**
  - **Observation:** This was slow for everyone. deque was fastest, then vector. list was the slowest by a huge amount.
  - **Conclusion:** For vector and deque, they have to shift a bunch of elements, which is slow. For list, the actual insert or delete is fast, but finding the random spot to do it is the real problem. It has to walk from the beginning every time. This walking is way slower than the element shifting that vector does.
- **IterateSum: Reading all elements**
  - **Observation:** Everyone was super fast. deque was the fastest, then vector, then list.
  - **Conclusion:** vector and deque keep their data all together in one nice block of memory. The computer loves this and can read it very fast. list has its data scattered all over the place, so the computer has to jump around to find the next piece. This is less efficient.
- **RandomAccess: Jumping to an element**
  - **Observation:** vector and deque were instant. list was terribly slow and just got worse and worse.
  - **Conclusion:** vector and deque can teleport to any element, like my_vector[100]. This is an instant operation. list cannot do this. To find the 100th element, it has to walk 100 steps from the start. To do this thousands of times is a disaster, and the time just explodes.

---

3. Summary

Here is what I learned from this:

1. **How time grows:**

    - **Good Growth:** Simple operations like PushBack or IterateSum grow in a predictable, straight line. If you double the data, the time roughly doubles.
    - **Bad Growth:** The problem operations like vector PushFront or list RandomAccess grow way faster. If you double the data, the time might quadruple or worse. These are the tests that would have run for hours.
2. **Why vector PushFront is so slow:**
    - It is just the wrong tool for the job. A vector is one solid block of memory. To put something at the front, it has to shift everything else over. deque and list do not. They just update a start pointer, which is instant.
3. **Which container is best?**
    - **vector**: Use this **most of the time**. It is the default. It is the fastest for reading, looping, and adding to the end. Just never use it to add things to the front.
    - **deque**: Use this if you need a **super vector**. It is almost as fast as vector at everything, but it also lets you add to the front instantly. Great for queues.
    - **list**: Use this **almost never**. It was the slowest in almost all my tests. Its only good trick is inserting in the middle, but that is only fast if you already know exactly where you are inserting. In my test, where it had to find the spot first, it was the worst.