I compared different functions of vectors and linked lists with sample sizes 10000, 100000, and 300000. The data is shown below.

|  |  |  |
| --- | --- | --- |
| Data Structure | Vector | Linked List |
| Insert | 10000: 0.204762 100000: 22.1695 300000: 196.582 | 0.28725 97.3942 1082.36 |
| Traverse | 5.71e-5 0.0006338 0.00203 | 0.000151 0.001314 0.0122 |
| Remove | 0.004187 0.354572 2.67932 | 0.00189 0.02324 0.0812 |
| Add value to the front | 0.00397 0.2548 2.78 | 0.000717 0.0107 0.0356 |
| Add value to the back | 0.000296 0.003092 0.00886 | 0.000697 0.0097 0.0399 |
| Remove value from the front | 0.00340 0.3354 2.85 | 0.000277 0.00288 0.0226 |
| Remove value from the back | 4.854e-5 0.000448 0.0014 | 0.000253 0.00364 0.0188 |
| Traverse unordered | 5.839e-5 0.0005868 0.00218 | 7.2947e-5 0.003149 0.00873 |

Based on the data, I found that if you want to add data at the front or remove the data at the front, using the linked list is a good idea. As for the vector, if you want to add data at the end or remove the data from the end, using the vector is a good idea. Besides, if you want to change any elements in the data structure, like sort or assign, or get elements from the data structure, like traverse or get, vector is still a good choice. Since the memory address for vectors is continuous, while that for linked lists is not, it is easier to get or change value in vectors than in linked lists.