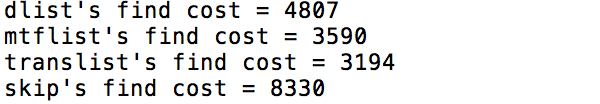
List Cost Analysis

# Goal and Experiment

The purpose of program 4 was to compare the performance of skip list with doubly linked list, move-to-front list (MTF), and transpose list. I implemented a skip list using four sided nodes organized into 6 levels to form a grid, the level on the grid were filled with a 50% chance of success and ended after the first failure. This allowed the skip list to have a find functions whose average complexity is bound to O(logn).

Using a statistical program, I tested the cost of each list by filling it with 1,000 integers of random numbers. I then picked 10 of those integers and accessed each item with a probability of the number’s index plus one divided by forty-five. Figure 1 shows the sample output from the statistics.cpp, these values are used to compare the cost of each list.

Figure 1: Sample Results From Statistical Program



# Results

Table 1 are the numerical cost for each list, the results are taken from the statistical driver. To better illustrate the change as size increases, the size of the list was increased by one hundred from one hundred to one thousand. The sample result from figure 1 is with a list size of ten.

Table 1: Numerical Cost Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size | Doubly Linked List Cost | MTF Cost | Transpose List Cost | Skip List Cost |
| 100 | 42598 | 3983 | 15907 | 11850 |
| 200 | 109399 | 4407 | 59198 | 17029 |
| 300 | 128972 | 4771 | 84393 | 21362 |
| 400 | 255291 | 6179 | 197849 | 30037 |
| 500 | 170241 | 5605 | 111832 | 33755 |
| 600 | 271747 | 6519 | 66406 | 28022 |
| 700 | 281449 | 7094 | 229966 | 46648 |
| 800 | 249114 | 6475 | 191918 | 53704 |
| 900 | 387179 | 7900 | 333576 | 46856 |
| 1000 | 587934 | 9172 | 524082 | 75821 |

Table 2 illustrates the cost for each list starting at the size being squared after each iteration. Table 2 is included to better show the trends as size approaches infinity.

Table 2: Numerical Cost Approaching Larger Size

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size | Doubly Linked List Cost | MTF Cost | Transpose List Cost | Skip List Cost |
| 10 | 4807 | 3590 | 3194 | 8330 |
| 20 | 12679 | 3598 | 3893 | 9691 |
| 40 | 25276 | 3769 | 5746 | 9354 |
| 80 | 27633 | 3784 | 6815 | 12971 |
| 160 | 60247 | 4171 | 27658 | 15084 |
| 320 | 118788 | 4814 | 66406 | 28022 |
| 640 | 356225 | 6690 | 293881 | 58737 |
| 1280 | 561521 | 8437 | 495587 | 116290 |
| 2560 | 1467763 | 18004 | 1401293 | 226399 |
| 5120 | 1880275 | 22321 | 1811494 | 636387 |

# Analysis

Figure 2 illustrates the values from Table 1 to better understand the results from the statistical program statistics.cpp. Doubly linked list was the costliest, transpose list following closely behind. Skip list was significantly cheaper, however MTF list was even lower being the least costly list of the four.

To illustrate that this trend continues as the size increases figure 3 shows the costs from a list size 10 to 5120. It is difficult to see the beginning values, but easy to see that as size increased doubly linked list and transpose list’s cost grows at a faster rate than skip list.

Figure 4 illustrates the comparisons from Figure 2 along with the values of the Big O notation. This is used to help understand how the costs compare in practice to the Big O notation. While it is difficult to see the smaller values the figure still illustrates the major trends.

# Conclusion

Skip list performed much better than doubly linked list and transposed list but was still slower than MTF list. Skip List’s use of levels significantly helped increase the cost of finding a value compared to the simpler version doubly linked list. Logically, skip list and doubly linked list still shares a worst case of O(n); however, in practice skip list on average performs much better than doubly-linked list.