**Report Program\_4**

Jun Hyung Park

**This Report Assumes:**

* Contains Four Algorithms (Move to Front, Transpose, Doubly Linked List, and Skip List).
* Data Size 3500, starts from 10 increase by 10 until 3500

**Goal:**

Implement Skip list to compare the performance and analyze the four algorithms’ find cost the four algorithms are Double Linked List’s, Transpose, MTFs, and Skip List. The ratio will be determined by using O(n^2), O(n), and O(log(n)).

**Experiment:**

The following experiment will contain the data size of starts from 10 and increase by 10 until the data hits 3500 and generate random item from index 0 to 29. Maximum size of pattern is defined as 10 and maximum sequence size is defined as 1000.  
Result out the cost of “Find” of each algorithm.

**Prove of Correctness of Skip List Algorithm**

* Result1.txt
* Result2.txt

will be provided separately.

**Result Observation from Result of Statistic.cpp:**

The result of “statistic.cpp” represent the find cost of each algorithms, which are Move to Front, Transpose, Doubly Linked List, and Skip List.

Figure 1, will represent the find cost of four algorithms

**Figure 1: Find Cost of Algorithms (Doubly Linked List, MTF List, Transpose List, and Skip List)**

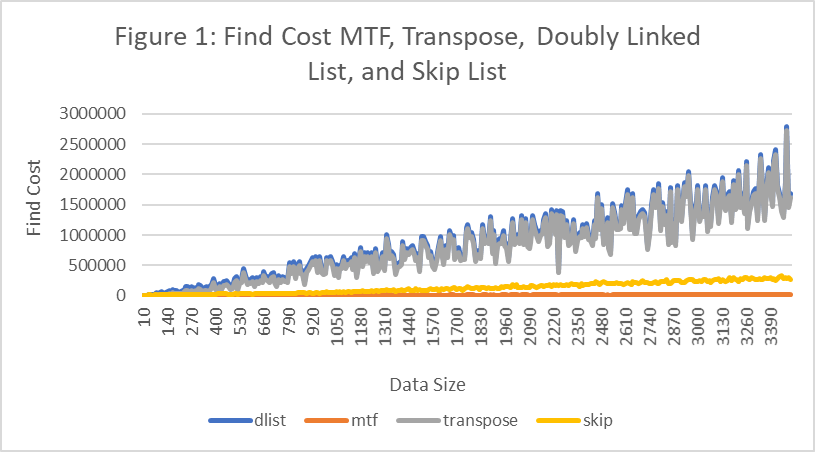
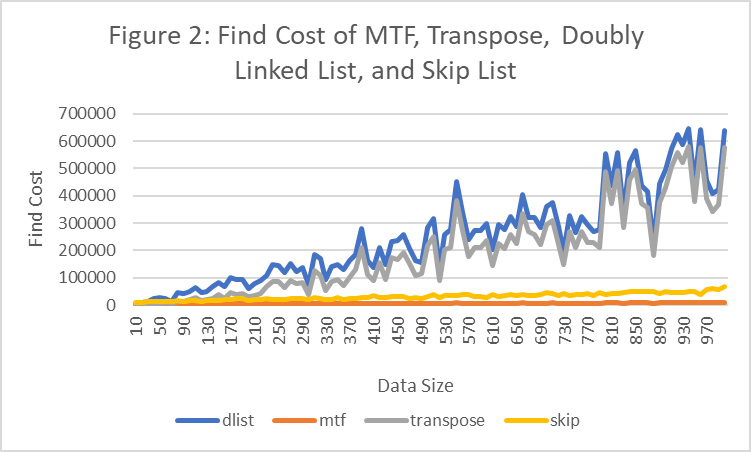
****

Figure 1 shows that the worst find cost among the four algorithms is Doubly Linked List, Transpose, and Skip List, and Move to Front List, with the data size of 3500. However, with figure 1, it is hard to visualize, when the data size is smaller. So, using figure 2, the smaller size of data will be provided.



The result of figure 2 shows clear that with the data size 1000, Doubly Linked List is still the worst, follow by Transpose, Skip List and MTF.

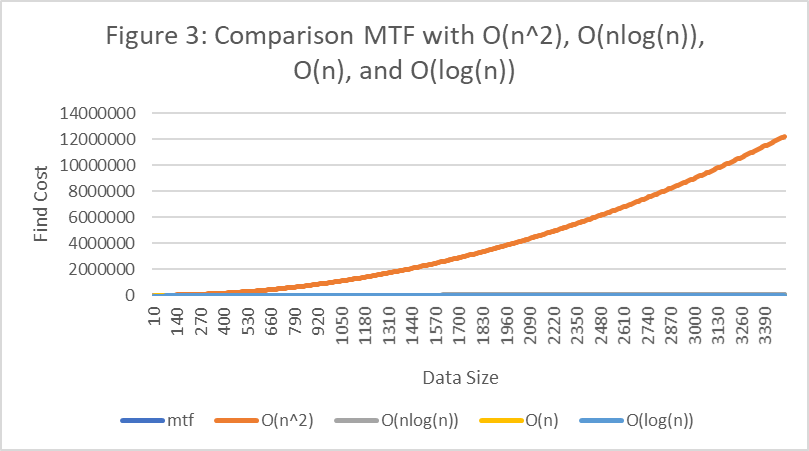
Therefore, I conclude that the worst find cost algorithm is Doubly Linked List, and Transpose, and Skip List, and MTF.

**Algorithm Analysis**

Each of the Algorithms, Move to Front, Transpose, Doubly Linked List, and Skip List’s cost will be checked through dividing with O(n^2), O(n), O(n\*log(n)), and O(log(n)), the ratio.

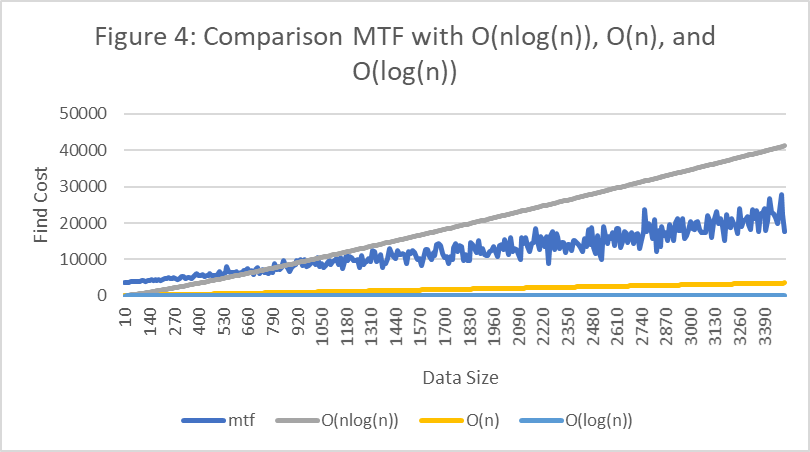
**Move to Front Analysis**

**The figure 3 will contain only Move to Front List with O(n^2, O(n), O(nlog(n)), and O(log(n))**



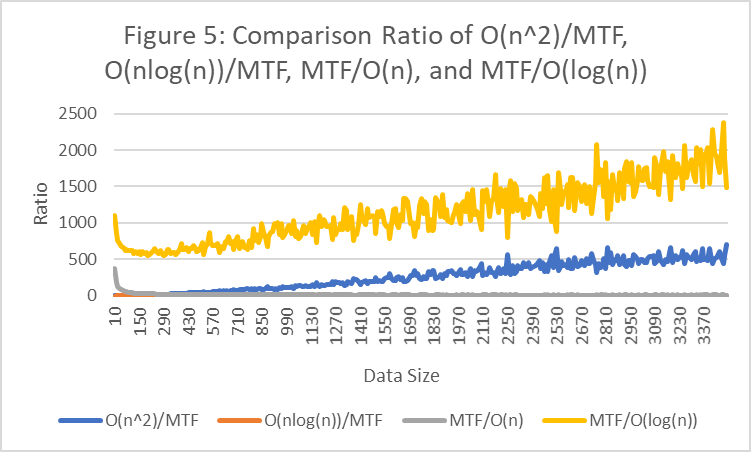
The result is clearly showing that the dominant factor is O(n^2), which makes hard to visualize other data. So, figure 4 will show all data except O(n^2).

**The Figure 4 will contain Move to Front List with O(n), O(nlog(n)), and O(log(n))**

****

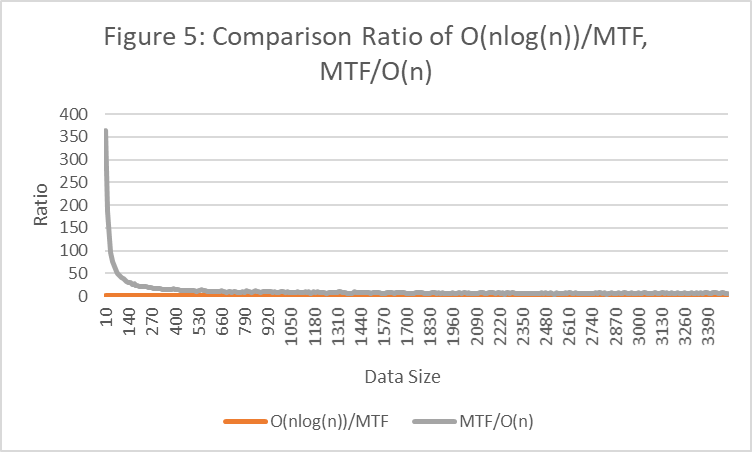
The result of Figure 3 and 4 shows that the upper bound for MTF List are O(n^2) and O(nlog(n)). The lower bound for MTF List are O(n) and O(log(n)). Therefore, Figure 5 will display the ratio of these Big-O with MTF List.

**The Figure 5 will contain ratio of O(n^2), O(n), O(nlog(n)), and O(log(n)) with MTF List.**



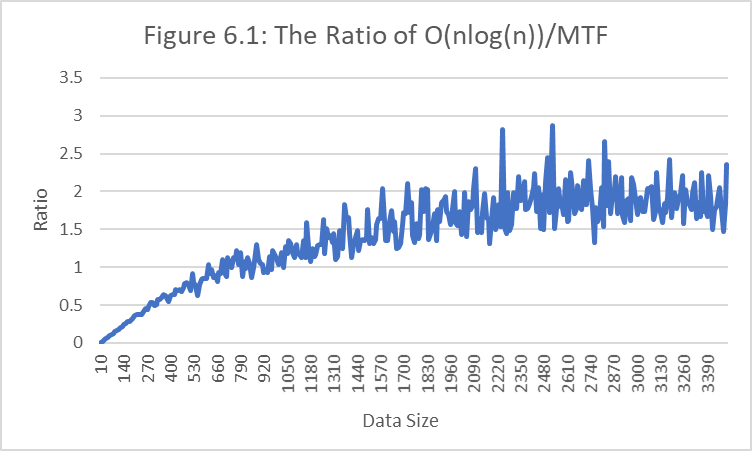
The result of figure 5 shows clearly that both O(n^2)/MTF and MTF/O(log(n)) are not constant. However, because of these two dominant factors, the ratio of O(nlog(n))/MTF and MTF/O(n) is not clear. Therefore, in figure 6, it will contain the all data except ratio of O(n^2)/MTF and MTF/O(log(n)).

**The figure 6 will contains ratio of MTF/O(nlog(n)) and MTF/O(n)**



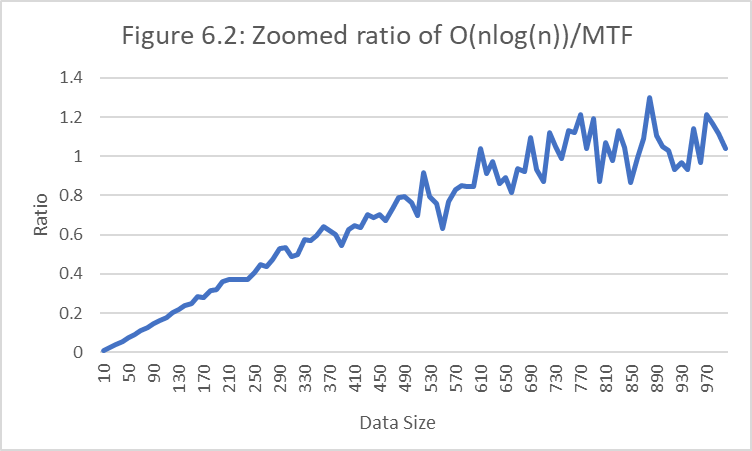
The result of figure 6 does not show any valuable information, therefore, figure 6.1 and figure 7 will contain each ratio visually.

**The figure 6.1 will contain only O(nlog(n))/MTF**

****

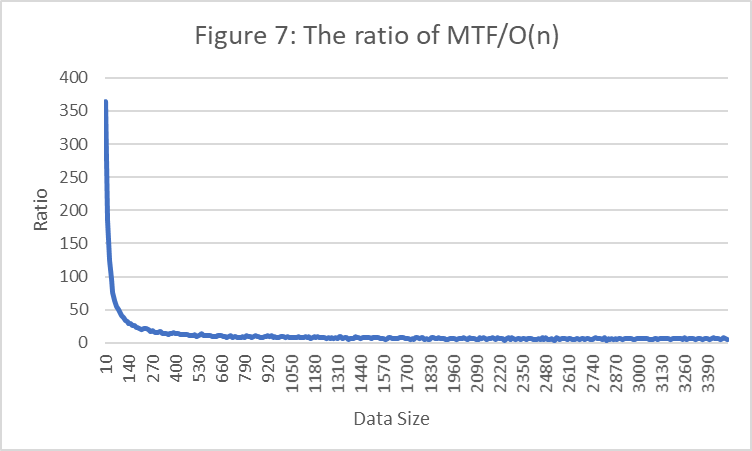
The result of figure 6.1 shows some how constant graph after certain amount of data. In order to make this clear, figure 6.2 will contain the same ratio but zoom in.

Figure 6.2 will contain zoomed O(nlog(n)) / MTF



The result of figure 6.2 shows that the ratio is linearly increase when the data size is about 1000.

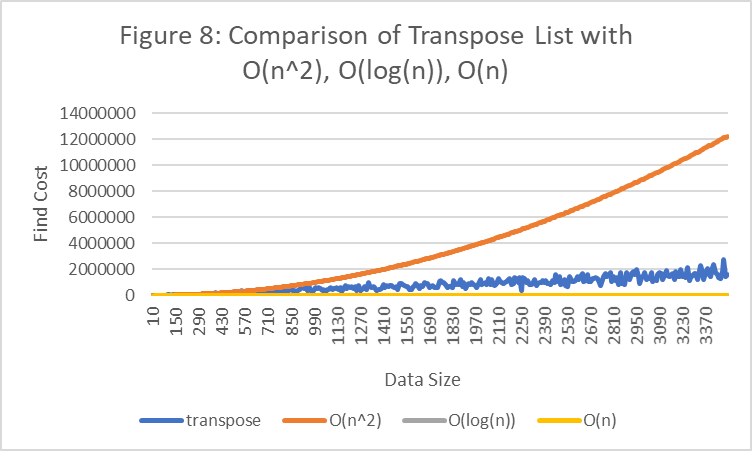
**The figure 7 will contains ratio of MTF/O(n)**

****

The result of Figure 7 shows that the ratio of MTF with O(n) is constant. To compare with another Big O, O(n) is the most constant graph for MTF List. Therefore, I conclude that the find cost of MTF is O(n).

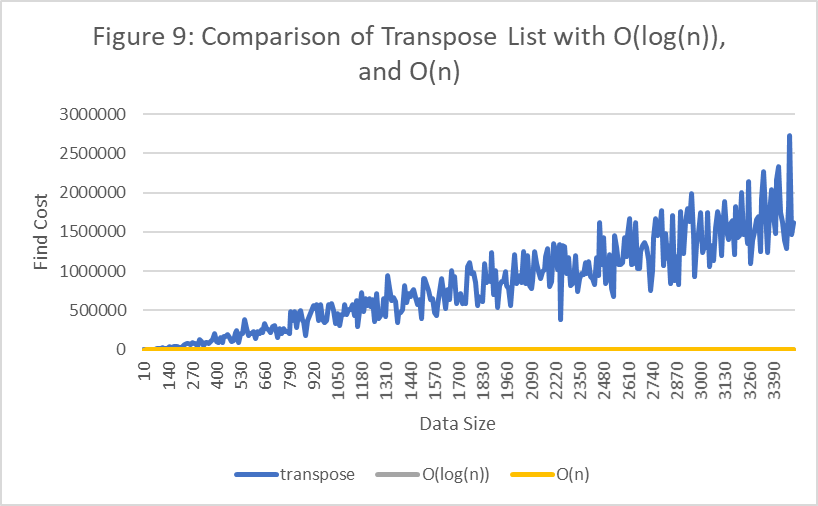
**Transpose Analysis**

**The figure 8 contains the comparison of Transpose with O(n^2), O(n), and O(log(n))**

****

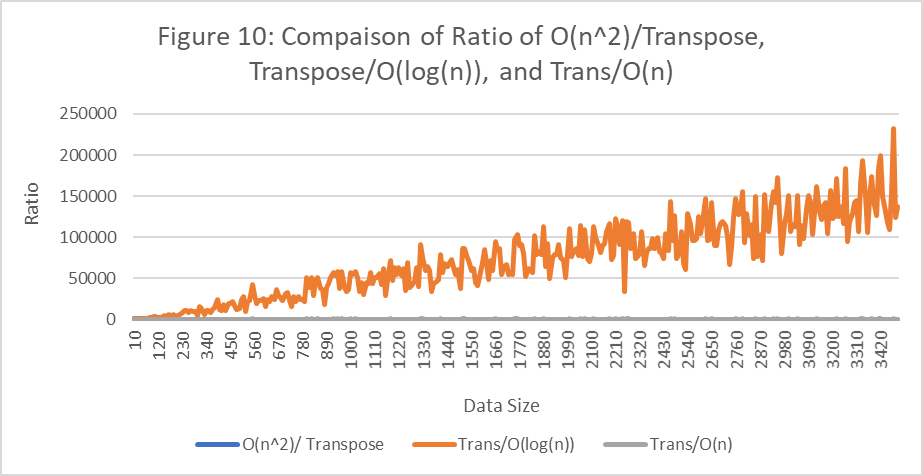
The result of figure 8 shows that the dominant factor is O(n^2) which make hard to visualize other data, therefore, figure 9 will contain Transpose, O(log(n)), and O(n).

**The figure 9 contains the comparison of Transpose with O(log(n)) and O(n)**



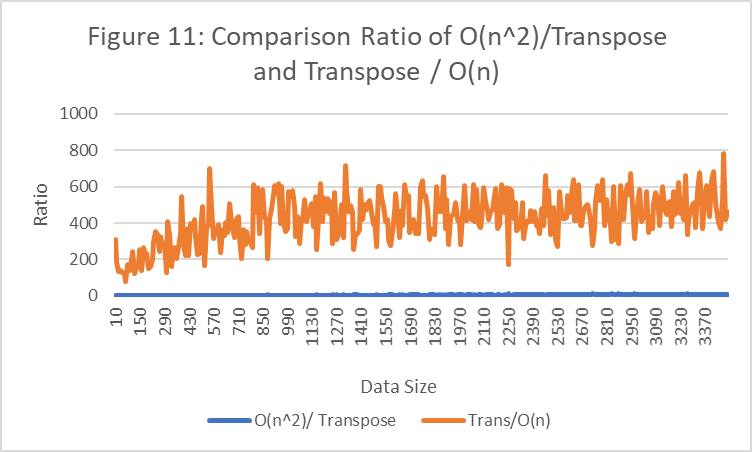
The result of figure 8 and 9 show that the upper bound of transpose is O(n^2) and lower bound of Transpose is O(log(n)) and O(n). Therefore, figure 10 will contains the ratio of Transpose dividing with O(n^2), O(log(n)) and O(n).

**Figure 10 contains the ratio of Transpose List with O(n^2), O(log(n)), and O(n)**

****

The result of figure 10 shows that the dominant factor, which is Trans/O(log(n)) makes hard to visualize the other data, therefore, figure 11, will only display Trans/O(n^2) and Trans/(O(n)).

**Figure 11 contains the ratio of Transpose List with O(n^2) and O(n)**

****

The result of figure 11 contains dominant factor which is Transpose / O(n), which makes hard to visualize the other data, therefore, figure 12 will contain only the ratio of O(n^2) / Transpose.

**Figure 12: Ratio of O(n^2) / Transpose**

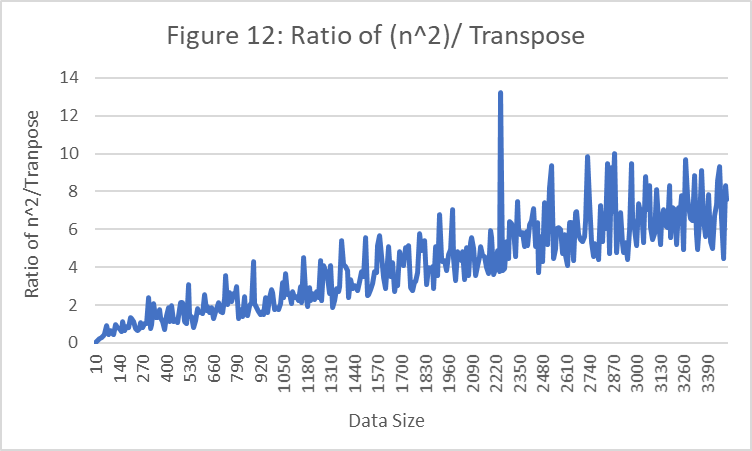
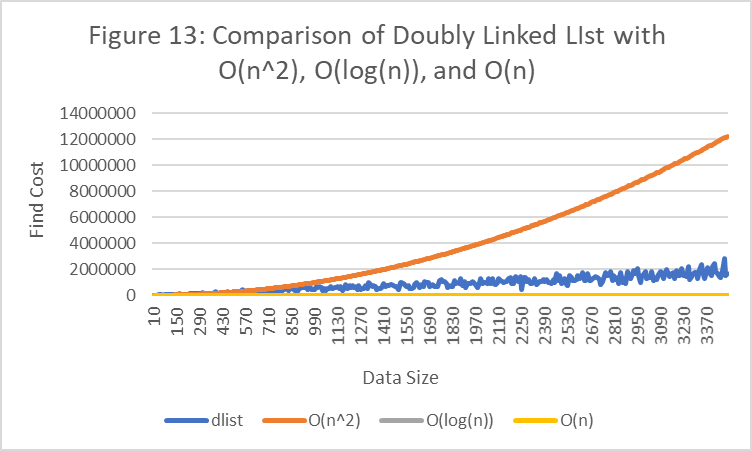
****

Figure 12 result shows that the graph is constantly increase and it contains higher slope compare to O(n), therefore, I conclude that find cost of Transpose is O(n).

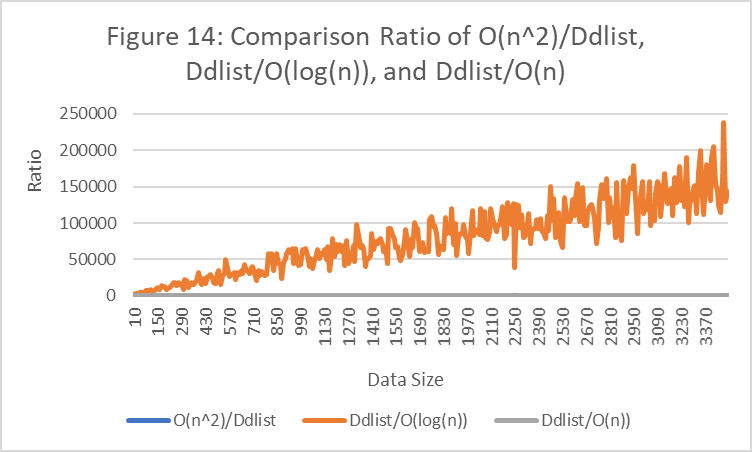
**Doubly Linked List Analysis**

**Figure 13 will contain the find cost of Doubly Linked List with O(n^2), O(n), and O(log(n))**

****

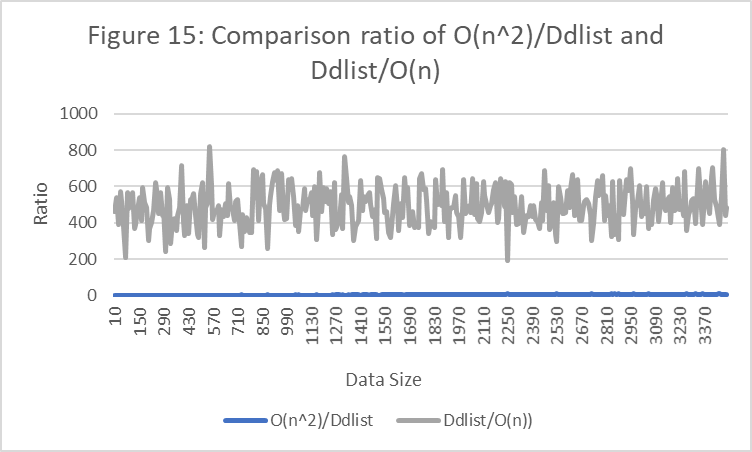
The result of figure 13 shows that the dominant factor is O(n^2) and since it is too dominant is hard to visualize under Doubly Linked List, however, it is clear that both O(log(n)) and O(n) are located below Doubly Linked List, so the lower bound of Doubly Linked List is O(log(n)) and O(n).

**Figure 14 contains the ratio of Doubly Linked List with O(n^2), O(log(n)), and O(n)**



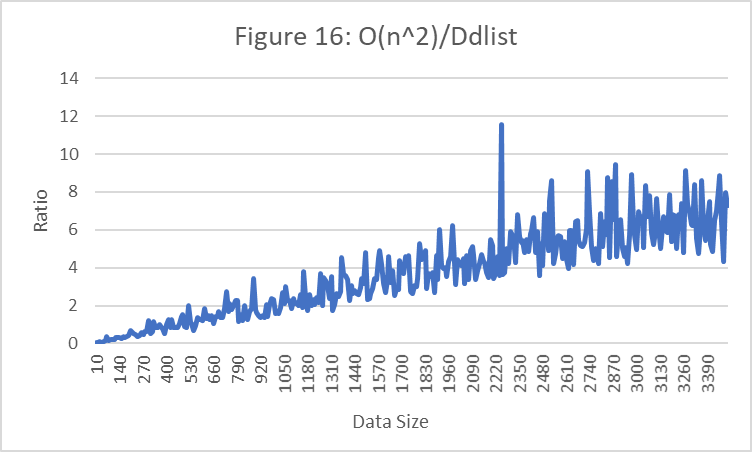
The result of figure 14 shows that the ratio Ddlist/O(log(n)) is not constant and it is dominant factor, which makes it hard to visualize other ratio data. Therefore, figure 15 will only contains the ratio of O(n^2)/Ddlist and Ddlist/O(n).

**Figure 15 contains the ratio of Doubly Linked List with O(n^2)/Ddlist and Ddlist/O(n)**

****

The result of figure 15 shows that the Ddlist/O(n)’s ratio is constant, however since the Ddlist/O(n) is dominant, it makes hard to visualize the ratio of O(n^2)/Ddlist. Therefore, figure 16 will only contains the ratio of O(n^2)/Ddlist.

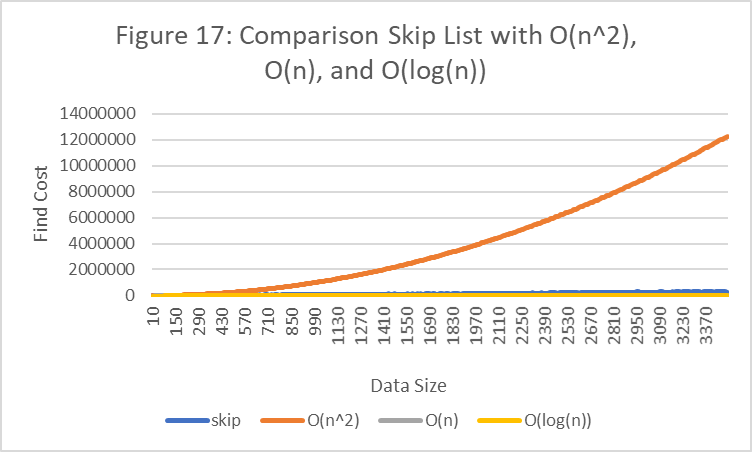
**Figure 16: Ratio of O(n^2)/Ddlist**



The result of figure 16 shows that the ratio of O(n^2)/Ddlist is not constant but linear, therefore, I conclude that the find cost of Doubly Linked List is O(n).

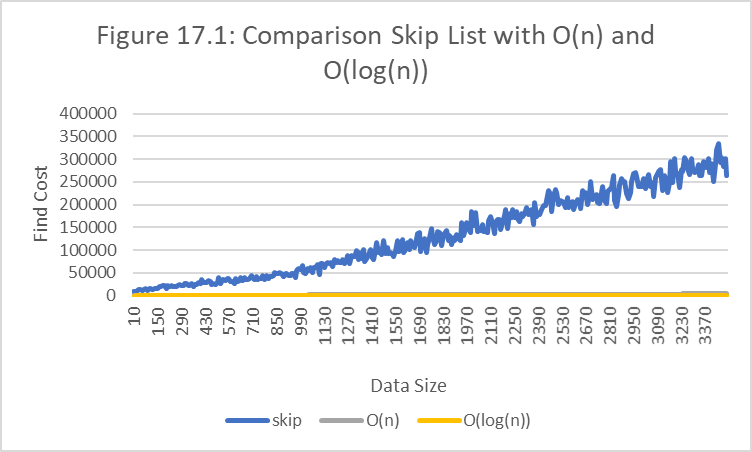
**Skip List Analysis**

**Figure 17 will contain the data that compare the find cost of Skip List with O(n^2), O(n), and O(log(n))**



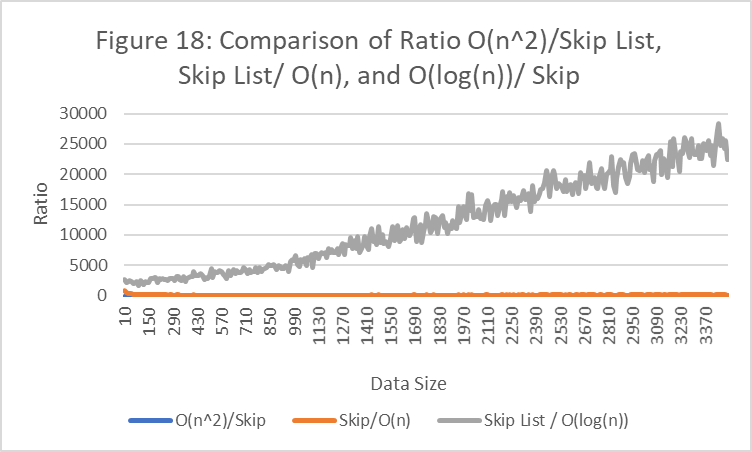
The result of figure 17 shows that the upper bound of Skip list is O(n^2) and because of the dominant factor O(n^2), it is hard to visualize what is below O(n^2), therefore, figure 17.1 will contain all data except O(n^2).

**Figure 17.1 shows comparison of Skip List with O(n) and O(log(n))**

****

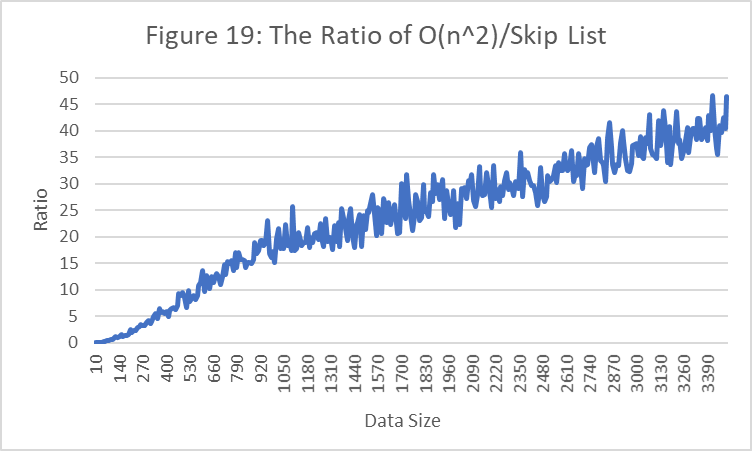
The result of figure 17.1 shows that both O(n) and O(log(n)) is lower bound of Skip List. Therefore, through figure 17 and 17.1, the upper bound of skip list is O(n^2) and lower bound of skip list is O(n) and O(log(n)). Now figure 18 will show the ratio of Skip List with O(n^2), O(n), and O(log(n)).

**Figure 18 shows that the ratio of Skip List with O(n), O(n^2), and O(log(n)).**



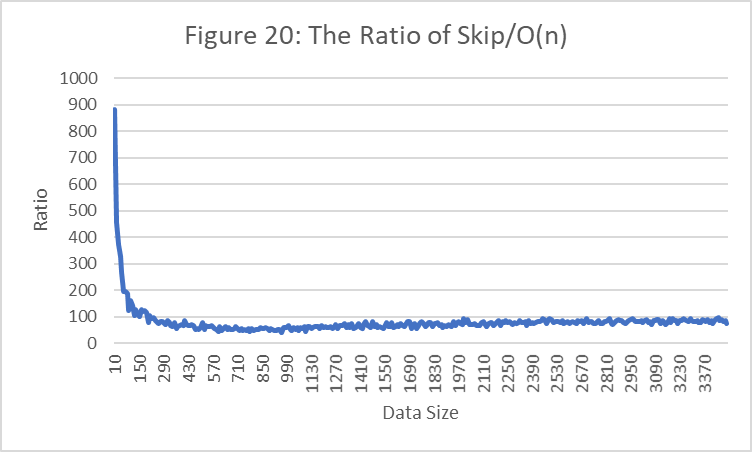
The figure 18 shows that all the ratio looks constant except Skip List/O(log(n)), therefore, evaluating the ratio of each algorithm separately is required. Figure 19 will only contain the ratio of O(n^2)/Skip List.

**Figure 19 shows that the ratio of Skip List with O(n^2) / Skip List**



The result of figure 19 shows that the ratio of O(n^2) / Skip List turned out to be not constant.

**Figure 20 contains only the ratio with Skip List / O(n)**

****

The result of figure 20 shows that the ratio of Skip List / O(n) is constant.

Therefore, I conclude that when the number of data size is 3500, find cost of Skip List is O(n).

**Conclusion:**

The purpose of this experiment was to analyze the find cost of four algorithms, which are MTF List, Transpose List, Doubly linked list, and Skip list. By having the data size of 3500 and comparing the four algorithms, the most efficient was MTF, Skip List, Transpose, and Doubly Linked List. By calculating the ratio of these four algorithms using O(n^2), O(n), O(log(n)), I concluded that find cost of MTF, Transpose, Doubly Linked List, and Skip List are O(n) as data increase.