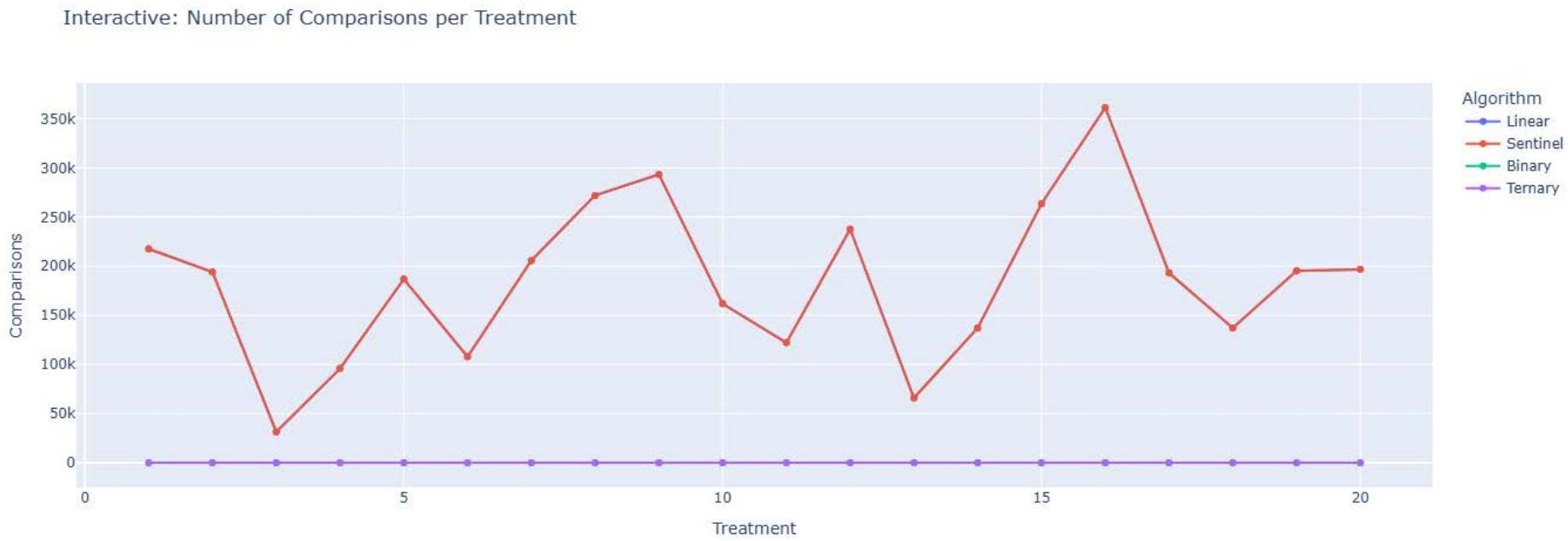
Question # 1: Perform the experiment to compare the efficiency of these four searching algorithms

```
Fetching data from URL:
https://raw.githubusercontent.com/mosomo82/COMP_SCI5501/refs/heads/main/Assignment/Assignment2_Searching_Algorithm/raw_data/words_alpha.txt
Successfully loaded 370104 words.
Generating 20 random search keys from the data...
Random keys selected:
['outsped', 'mottlement', 'bemaddening', 'eel', 'midshipman', 'explosion', 'nonparental', 'rhina', 'skiv',
'joshes', 'garn', 'pyramiding', 'confetto', 'hermele', 'recoverer', 'volutoid', 'moonite', 'hereunder',
'multipass', 'nanocephaly']
Running treatments...
Experiment complete.

--- Comparative Analysis for Each Treatment ---

Execution Time (µs) per Treatment:
Algorithm    Binary    Linear  Sentinel   Ternary
Treatment
1          42174.82  34379.48  34379.48  45450.45
2          47467.71  31974.08  31974.08  47246.69
3          48857.21   5464.79   5464.79  43313.03
4          46519.76  13313.06  13313.06  47062.40
5          41558.74  32471.66  32471.66  40188.07
6          22663.83   7822.28   7822.28  23235.08
7          22923.23  10722.88  10722.88  22775.41
8          22733.69  13965.85  13965.85  24405.48
9          22857.90  14860.63  14860.63  24148.46
10         24402.62   8542.54   8542.54  22716.05
11         22671.94   6145.95   6145.95  24322.99
12         22478.58  12693.64  12693.64  22404.67
13         23543.12   3528.59   3528.59  22795.44
14         22459.51   8964.78   8964.78  25196.55
15         22847.89  15307.19  15307.19  22499.80
16         36891.70  19360.78  19360.78  23051.26
17         22850.28  13194.56  13194.56  23035.53
18         22845.03   7566.69   7566.69  23083.69
19         22572.99  10452.03  10452.03  22771.84
20         22409.92  10252.71  10252.71  22709.61
```

Number of Comparisons per Treatment:

Algorithm	Binary	Linear	Sentinel	Ternary
Treatment				
1	19	217544	217544	32
2	17	194114	194114	32
3	18	31346	31346	28
4	17	95889	95889	34
5	15	186816	186816	32
6	18	107911	107911	34
7	16	205758	205758	31
8	18	271974	271974	34
9	18	293413	293413	31
10	18	161812	161812	34
11	18	122203	122203	32
12	19	237586	237586	32
13	17	65874	65874	34
14	17	137148	137148	32
15	18	263560	263560	35
16	17	361293	361293	34
17	18	193062	193062	34
18	18	137096	137096	34
19	19	195235	195235	31
20	17	196762	196762	34

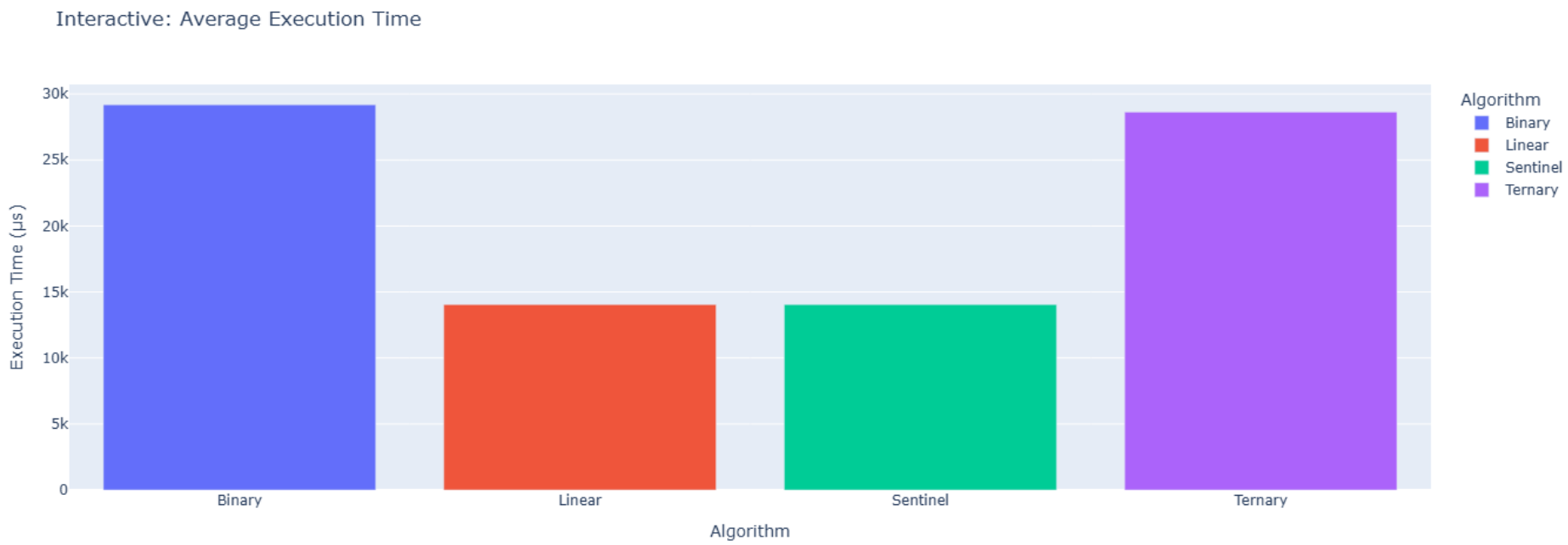
```
Generating interactive line charts...
```

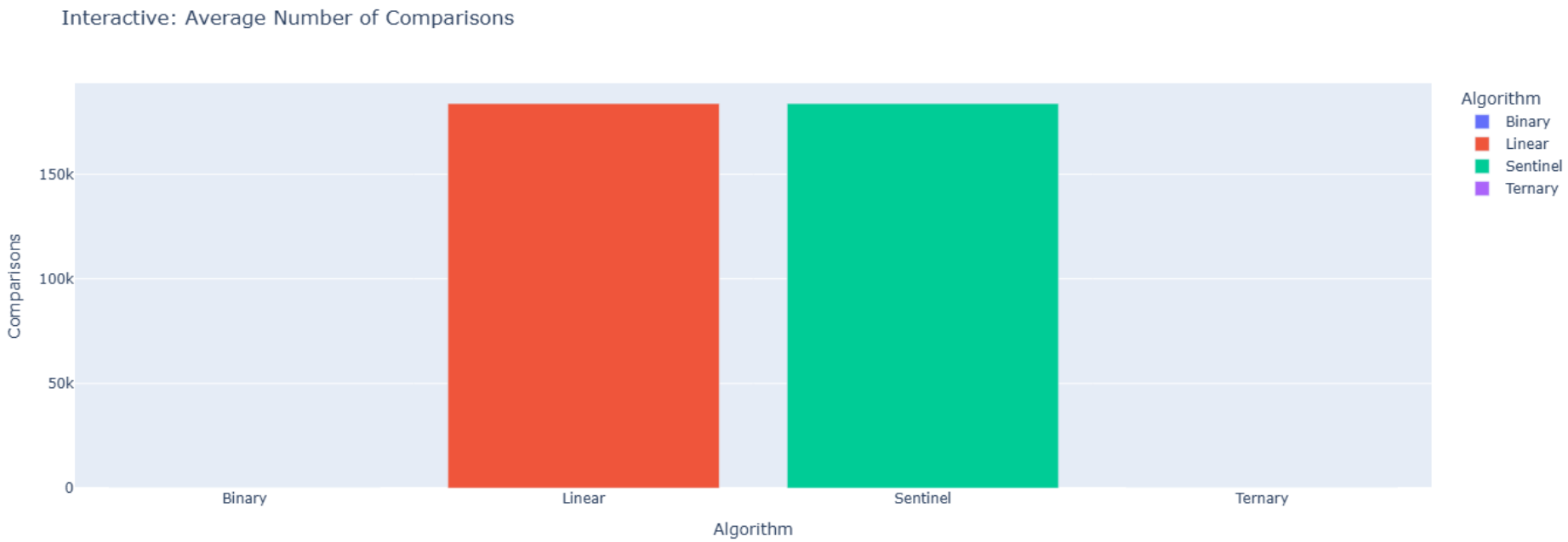


Comparisons per TreatmentTreatmentComparisons

```
--- Average Performance Analysis ---
           Execution Time (µs)  Comparisons
Algorithm
Binary                29186.52         17.6
Linear                14049.21     183819.8
Sentinel              14049.21     183819.8
Ternary               28620.62         32.7
```

```
Generating interactive bar charts...
```





iii. Based on the observation:

- **Execution time:** Linear search and sentinel search have way lower average execution times compared to binary search and ternary search. The reason is the cost of sorting data are included for binary and ternary search. By its mean, the cost of sorting step is expensive and it dominate the search time for this experiment setup.
- **Comparisons:** Binary and ternary searches have very lower average number of comparisons when comparing to linear and sentinel search. The reason is the algorithm of binary and ternary search. They divide the search space in the large **sorted** dataset.

In summary, binary and ternary searches are much more efficient in terms of the number of comparisons for sorted data.

Question # 2: Perform the experiment to compare the performance of 2D list over Linear List by executing only Linear Search.

```
Fetching data from
https://raw.githubusercontent.com/mosomo82/COMP_SCI5501/refs/heads/main/Assignment/Assignment2_Searching_Algorithm/raw_data/words_alpha.txt
Successfully loaded 370104 words.

Running 10 search treatments...

--- Comparison Results per Treatment ---
   Treatment        Search Key  Comparisons (1D List)  Comparisons (2D List)
0          1       stagflation                 302326                  26353
1          2           mealily                 181715                   5123
2          3          grainery                 128657                   7897
3          4  literalistically                 173569                   6979
4          5        cyanoauric                  56848                  13018
5          6        allometric                   8879                   8879
6          7           unparch                 347444                  13888
7          8       subpeduncle                 308787                  32814
8          9           piniest                 237067                  14530
9         10         jalalaean                 160205                    407

--- Average Number of Comparisons ---
Comparisons (1D List)    190549.7
Comparisons (2D List)     12988.8

dtype: float64
```

4. Based on the comparison results for linear search on the 1D list versus the 2D list. This shows that choosing the right data structure choice can greatly impact the algorithm performance. Structuring the data into a 2D list based on the first letter eliminates unnecessary search space and focuses on searching on the smaller sub-list. As a result of the above, a 2D list only requires an average of 13,000 comparisons, which is way lower than that of a 1D list having an average of 190,000 comparisons.

https://github.com/mosomo82/COMP_SCI5501/blob/main/Assignment/Assignment2_Searching_Algorithm/src/Assignment2_5501_Searching_Algorithm.ipynb

https://colab.research.google.com/github/mosomo82/COMP_SCI5501/blob/main/Assignment/Assignment2_Searching_Algorithm/src/Assignment2_5501_Searching_Algorithm.ipynb