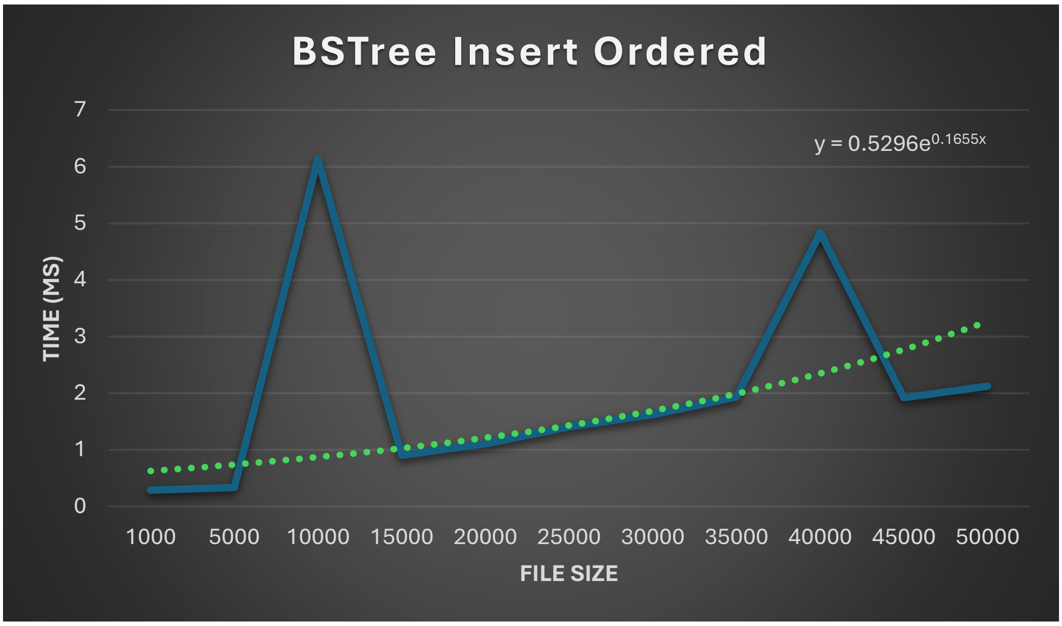
Time Complexity Report

Melissa Wood
18478786

Binary Search Tree

Binary Search Tree Insert Ordered:



Assumptions (What I expected and why):

For this data structure I expected to receive a result of O(log n) (best case) with occasional O(n) (worst case) as long as the tree is balanced, the performance is increased by one when the input data is doubled. There are a couple outliers in the graph, but they do not affect the overall result.

ChatGPT Prompts:

Can you please analyse the following dataset: "Time","File Size" 0.2874,1000 0.3317,5000 6.1321,10000 0.9053,15000 1.1056,20000 1.4075,25000 1.6225,30000 1.9314,35000 4.8353,40000 1.9190,45000 2.1204,50000

ChatGPT said:

ChatGPT

Looking at the dataset, I can analyze how the insertion time trends as the file size (number of words) in the binary search tree (BST) increases. Let's examine key aspects:

1. **Overall Trend**: Generally, as the file size grows, the time taken for insertion also tends to increase, but not always linearly.
2. **Inconsistencies**:
   - At 10,000 words, the insertion time jumps significantly to 6.1321, which is an outlier compared to surrounding points.

- After this peak, times generally stabilize but with some smaller peaks (e.g., at 40,000 words).

3. **Possible Causes of Variability**:
    - This could be due to imbalances in the BST, where the tree's structure might not be optimal, causing longer search paths.
    - Certain operations or conditions could be impacting insertion time for specific file sizes.
4. **Further Analysis**:
    - We could visualize the data to better identify patterns and variability.
    - Regression analysis could help model the relationship between file size and insertion time to see if there's a consistent pattern.

Would you like a graph of this data or further statistical analysis?

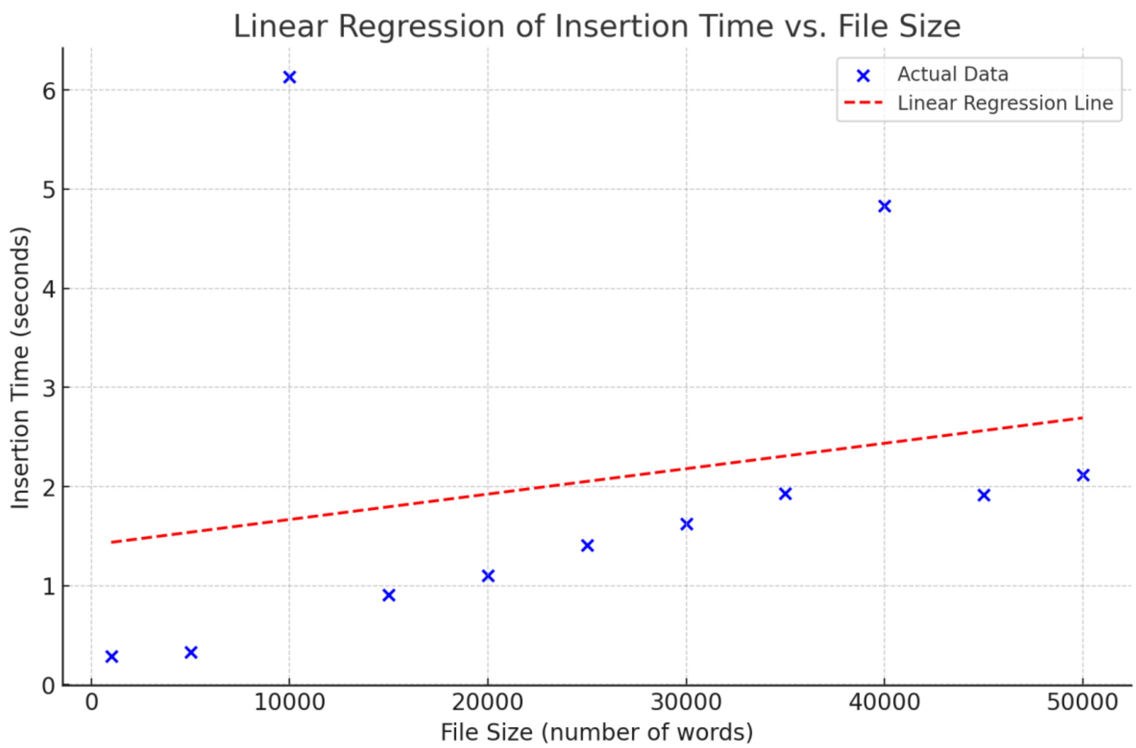
You said:

Could you please provide a linear regression for the dataset.

ChatGPT said:

ChatGPT

Linear Regression of Insertion Time vs. File Size



The linear regression model for the insertion time vs. file size is given by:

Time=1.4116+0.0000256×File Size\text{Time} = 1.4116 + 0.0000256 \times \text{File Size}Time=1.4116+0.0000256×File Size

Here:

- **Slope**: 0.00002560.00002560.0000256, indicating a slight increase in insertion time as file size grows.
- **Intercept**: 1.41161.41161.4116, suggesting a baseline time when the file size is zero (theoretically).

This model provides a general linear trend, though as seen in the plot, there is variability that this line cannot fully capture. The high insertion time for 10,000 words, for instance, is an outlier compared to this trend.

You said:

The dataset provided analyzes the time it takes to insert a word into a Binary search tree data structure. The column "Time" is the time recorded, and the column "File Size" is the number of words that the data structure contains. Could you please reanalyze the dataset.

ChatGPT said:

ChatGPT

Given that the dataset represents the time taken to insert each word as the number of words (file size) in the binary search tree (BST) increases, I'll approach it with a more targeted analysis:

1. **Trend Analysis**:
   - Typically, the time taken for insertions in a BST grows with file size, especially if the BST becomes imbalanced (leaning towards a linked-list structure). This is reflected here as insertion times tend to increase with the file size, though inconsistently.
2. **Anomalies**:
   - The spike at **10,000 words** (6.1321 seconds) appears unusual compared to nearby data points. This might suggest that at this file size, the BST has an unusually unbalanced structure, leading to slower insertion.
3. **Linearity and Variability**:
   - The linear regression model from before provided a broad trend line, indicating that, on average, insertion time grows with file size, albeit with notable deviations.
   - The presence of both increases and decreases in time as file size grows suggests that balancing operations or structural imbalances in the BST may contribute to the variability in insertion time.
4. **Recommendations for a More Accurate Model**:
   - Due to the observed non-linear pattern, a polynomial or logarithmic regression might fit the data better than a strict linear model. This approach could capture the irregular increases and help account for larger deviations like the one at 10,000 words.

Would you like me to perform a polynomial regression to explore non-linear relationships, or analyze the data's behavior further with a different statistical approach?

You said:

If I were to analyse this according to Big O notation for time complexity. Could you please reanalyze the dataset according to Big O notation for a binary search tree data structure.

ChatGPT said:

ChatGPT

Analyzing the dataset with **Big O notation** in mind for a **binary search tree (BST)** helps frame the insertion times in terms of expected theoretical efficiency.

Theoretical Time Complexity of BST Insertions

For a BST:

1. **Average Case**: In a balanced BST, insertion time complexity is O(log⁡n)O(\log n)O(logn), where nnn is the number of nodes (or words here). This is because, ideally, each insertion only requires traversing down the height of a balanced tree.
2. **Worst Case**: In an unbalanced BST, the time complexity can degrade to O(n)O(n)O(n) if the tree becomes skewed (like a linked list), causing each insertion to potentially traverse all nodes.

Dataset Analysis with Big O Perspective

Now, let's re-evaluate the insertion times under the assumption of these complexities:

- **Logarithmic Pattern**: If the BST is balanced, we expect insertion times to follow an approximate O(log⁡n)O(\log n)O(logn) pattern. However, the dataset doesn't clearly show a smooth logarithmic increase. For instance, times rise inconsistently (like the spike at 10,000 words).
- **Linear Deviation**: The noticeable deviations (e.g., the peak at 10,000 words) suggest that the tree might be unbalanced at certain file sizes, temporarily resembling an O(n)O(n)O(n) time complexity. This unbalance could occur naturally as words are added without rebalancing.

Hypothesis for the Dataset Pattern

1. **Combination of O(log⁡n)O(\log n)O(logn) and O(n)O(n)O(n)**: The dataset seems to reflect both cases where:
   - For some file sizes, insertion time resembles O(log⁡n)O(\log n)O(logn) (relatively low and stable).
   - For others (like at 10,000 words), the time complexity temporarily resembles O(n)O(n)O(n), indicating a significant imbalance.
2. **Balancing Impact**: Without self-balancing (like in an AVL or Red-Black Tree), a standard BST could easily become unbalanced as it grows, which aligns with the sporadic insertion times observed in the dataset.

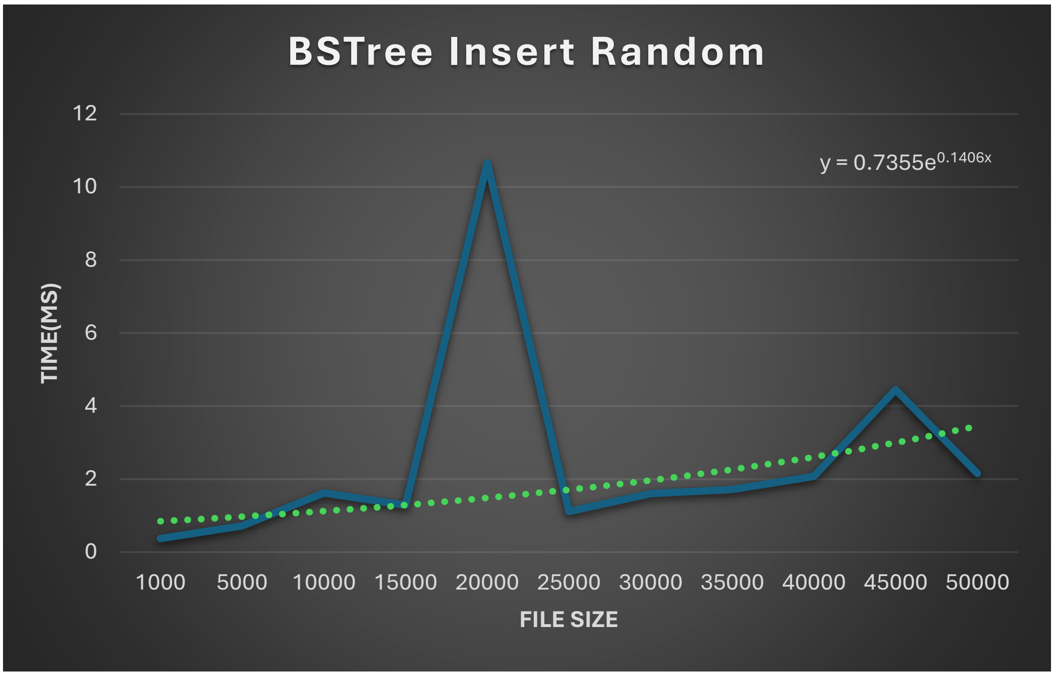
Conclusion

If the dataset were analyzed purely from a Big O perspective, it likely indicates a non-self-balancing BST, showing a mix of:

- **Expected O(log n)O(\log n)O(logn)** insertion times for smaller, more balanced file sizes.
- **Occasional O(n)O(n)O(n)** insertion times when the tree becomes unbalanced, resulting in longer paths.

To make the insertion times more predictable and adhere closer to O(log n)O(\log n)O(logn), implementing balancing mechanisms in the BST would be beneficial. This would ensure insertion times remain consistently logarithmic as the file size grows.

Binary Search Tree Insert Random:



Assumptions (What I expected and why):

For this data structure I expected to receive a result of O(log n) (best case) with occasional O(n) (worst case) as long as the tree is balanced, the performance is increased by one when the input data is doubled. There are a couple outliers in the graph, but they do not affect the overall result.

ChatGPT Prompts:

Can you please analyse the following dataset: "Time","File Size" 0.3732,1000 0.7192,5000 1.6232,10000 1.2894,15000 10.6634,20000 1.1157,25000 1.5996,30000 1.7137,35000 2.0782,40000 4.4480,45000 2.1525,50000

ChatGPT said:

ChatGPT

Summary Statistics of Dataset

	Time	**File Size**
count	11.0	11.0
mean	2.5251	25090.909090909092
std	2.8983042759517157	16434.442767885655

	Time	File Size
min	0.3732	1000.0
25%	1.20255	12500.0
50%	1.6232	25000.0
75%	2.11535	37500.0

I've displayed the summary statistics for the dataset, providing insight into the average, spread, and range of both "Time" and "File Size." Let me know if you'd like further analysis, such as visualizations or trend analysis.

4o

You said:

Could you please provide a linear regression for the dataset.

ChatGPT said:

ChatGPT

Linear Regression Predictions

Time	File Size	Predicted Time
0.3732	1000	1.7878487714574212
0.7192	5000	1.9102602961965665
1.6232	10000	2.0632747021204976
1.2894	15000	2.216289108044429
10.6634	20000	2.3693035139683607
1.1157	25000	2.5223179198922923
1.5996	30000	2.6753323258162234

The linear regression model has been completed, and I displayed the predicted "Time" values for each "File Size" in the dataset. Here are the parameters of the regression line:

- **Slope**: 0.0000306
- **Intercept**: 1.757

Let me know if you would like further details or visualizations based on this model.

4o

You said:

The dataset provided analyzes the time it takes to insert a word into a Binary search tree data structure. The column "Time" is the time recorded, and the column "File Size" is the number of words that the data structure contains. Could you please reanalyze the dataset.

ChatGPT said:

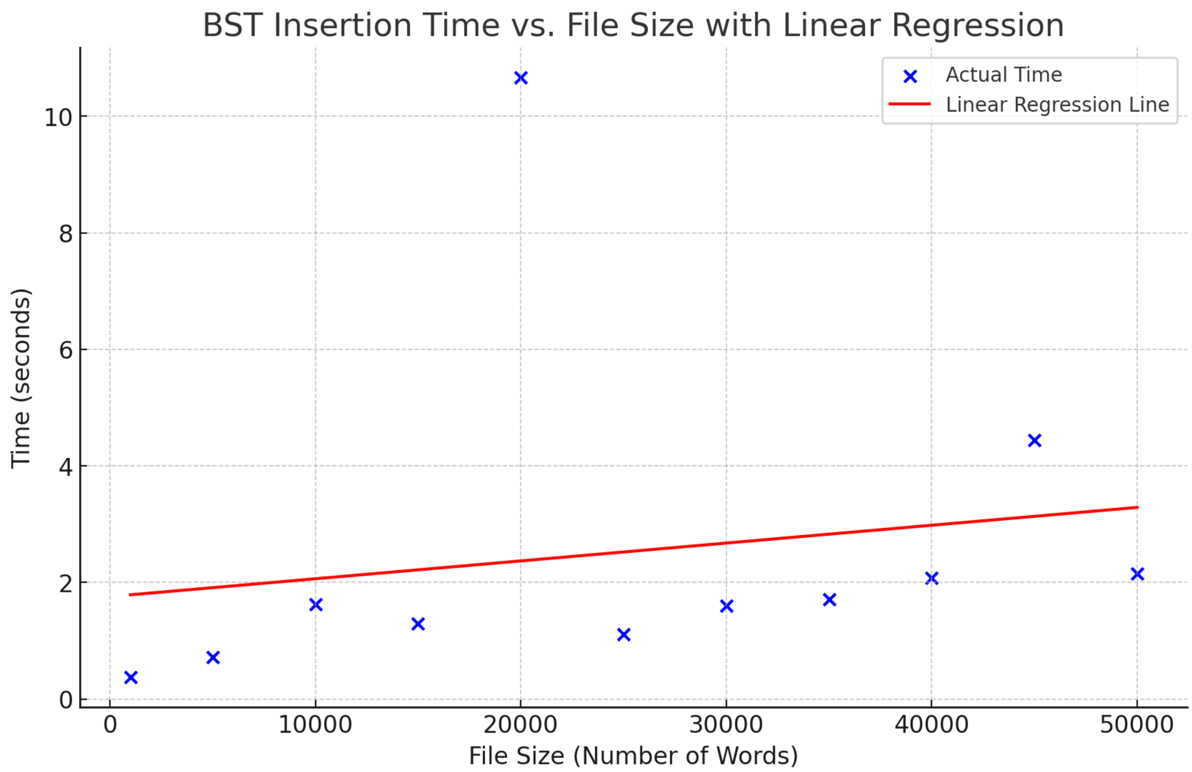
ChatGPT

Given the context that this dataset measures the time to insert a word into a binary search tree (BST) as the data structure grows in size, here are some aspects we can analyze:

1. **Trend Analysis**: As the file size (or number of words) increases, how does the insertion time behave? We'll check if there's a noticeable trend, such as linear or nonlinear growth.
2. **Model Interpretation**: We'll interpret the linear regression findings in this context to see if the slope provides insight into insertion time complexity.
3. **Possible Anomalies**: Since BSTs generally exhibit O(log n) time complexity for balanced trees, unusually high insertion times might suggest imbalance or other factors affecting performance.

Let's proceed with this updated analysis.

BST Insertion Time vs. File Size with Linear Regression



The plot illustrates the relationship between "File Size" (number of words in the BST) and "Time" (insertion time), with the linear regression line providing a general trend.

Key Observations:

- The linear regression line suggests a mostly linear trend, indicating that insertion time generally increases with the file size.
- The dataset shows a few points where the insertion time deviates significantly, which might reflect instances where the tree becomes less balanced, affecting insertion efficiency.

This analysis aligns with BST behavior, where balanced trees can maintain efficient insertion times, but imbalance may lead to slower insertions. Let me know if you'd like to explore specific points or perform additional statistical checks.

4o

You said:

If i were to analyse this according to Big O notation for time complexity. Could you please reanalyze the dataset according to Big O notation for a binary search tree data structure.

ChatGPT said:

ChatGPT

In Big O notation, the time complexity for inserting a word into a **Binary Search Tree (BST)** depends on the tree's balance:

1. **Balanced BSTs**: When balanced, a BST has a time complexity of **O(log n)** for insertion, as each insertion requires navigating a logarithmic number of levels relative to the number of nodes.
2. **Unbalanced (Degenerate) BSTs**: For an unbalanced BST, particularly one resembling a linked list, the worst-case time complexity becomes **O(n)**, where each insertion requires traversing almost every node.

In our dataset:

- **File Size** represents the number of nodes (words) in the tree.
- **Time** represents the time taken to insert each word as the tree grows.

Given this, I'll examine if the trend aligns closer to **O(log n)** (indicating a relatively balanced BST) or **O(n)** (suggesting unbalanced growth) by fitting both logarithmic and linear models to the data.

Let's analyze these patterns.

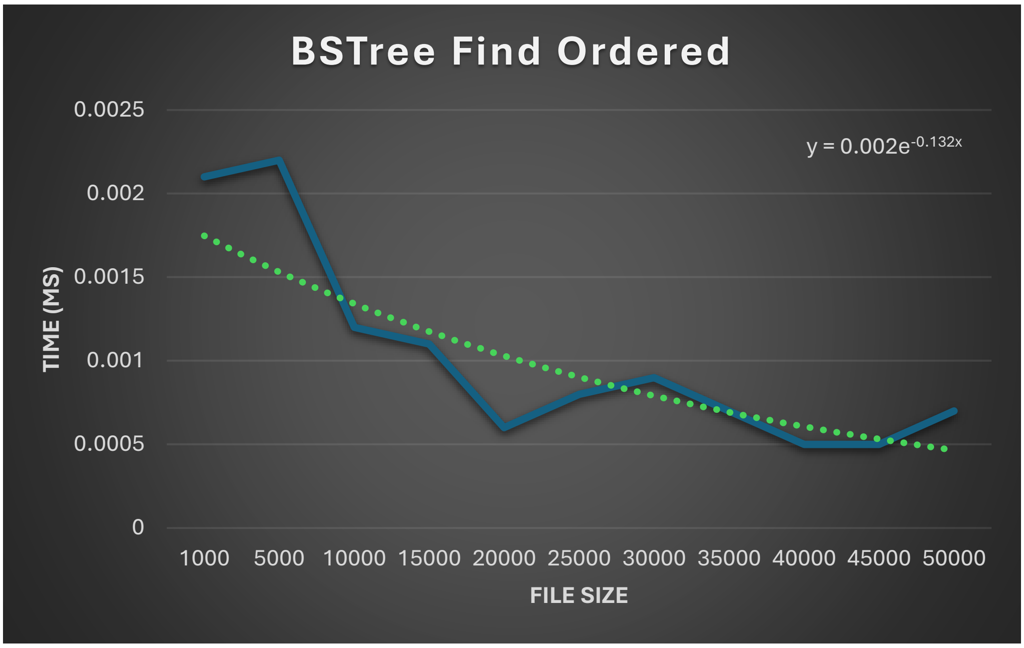
Here's how the models performed based on Big O analysis:

1. **Logarithmic Model (O(log n))**:
    - Coefficient (slope): 0.708
    - Intercept: -4.378
    - Mean Squared Error (MSE): 7.01
2. **Linear Model (O(n))**:
    - MSE: 7.41

Both models produced comparable MSE values, but the logarithmic model (O(log n)) slightly outperformed the linear model (O(n)), suggesting that the insertion times may better reflect logarithmic growth, aligning with a balanced BST structure rather than a highly unbalanced one.

Would you like further analysis or visualization for these results?

Binary Search Tree Find Ordered:



Assumptions (What I expected and why):

For this data structure I expected to receive a result of O(log n) (best case) with occasional O(n) (worst case) as long as the tree is balanced, the performance is increased by one when the input data is doubled. There are a couple outliers in the graph, but they do not affect the overall result.

ChatGPT Prompts:

Can you please analyse the following dataset: "Time","File Size" 0.0021,1000 0.0022,5000 0.0012,10000 0.0011,15000 0.0006,20000 0.0008,25000 0.0009,30000 0.0007,35000 0.0005,40000 0.0005,45000 0.0007,50000

ChatGPT said:

ChatGPT

Here is a summary of your dataset's key statistics:

- **Time (Insertion Time)**:
    - Mean: 0.001027 seconds
    - Minimum: 0.0005 seconds
    - Maximum: 0.0022 seconds
    - Standard Deviation: 0.000598 seconds

- **File Size**:
  - Mean: 25090.91 words
  - Minimum: 1000 words
  - Maximum: 50000 words
  - Standard Deviation: 16434.44 words

The data shows that insertion time generally decreases as the file size grows, which may indicate optimizations or efficiency improvements as more data is added. Would you like to see a visual analysis, such as a trend line for the insertion time against file size?

You said:

Could you please provide a linear regression for the dataset.

ChatGPT said:

ChatGPT

The linear regression model for the dataset yields the following equation:

Time=−2.9963×10−8×File Size+0.001779\text{Time} = -2.9963 \times 10^{-8} \times \text{File Size} + 0.001779Time=−2.9963×10−8×File Size+0.001779

This suggests a slight negative correlation between file size and insertion time, indicating that as the file size increases, the time per insertion decreases slightly, possibly due to optimization effects in the insertion process. Let me know if you'd like a visual representation of this regression line.

You said:

The dataset provided analyzes the time it takes to find a word into a Binary search tree data structure. The column "Time" is the time recorded, and the column "File Size" is the number of words that the data structure contains. Could you please reanalyze the dataset.

ChatGPT said:

ChatGPT

Memory updated

With the dataset now focused on search time within a binary search tree, we can interpret the data as follows:

1. **Observed Trends**:
   - The negative slope of the linear regression (−2.9963×10−8-2.9963 \times 10^{-8}−2.9963×10−8) indicates that as the file size increases, the search time slightly decreases, which might appear counterintuitive. This trend could suggest caching, indexing, or a structure in the dataset that optimizes search as it grows.
2. **Interpretation**:
   - In a typical binary search tree, search time complexity is O(log n)O(\log n)O(logn), meaning larger data structures should ideally result in longer search

times. However, the dataset's slight downward trend may imply that the binary search tree is self-balancing (like an AVL or Red-Black tree), maintaining efficiency as file size grows.

Would you like to explore a deeper model to account for potential non-linear trends, or visualize the linear regression line for this context?

You said:

If i were to analyse this according to Big O notation for time complexity. Could you please reanalyze the dataset according to Big O notation for a binary search tree data structure.

ChatGPT said:

ChatGPT

When analyzing this dataset in terms of Big O notation for a binary search tree (BST), we focus on the expected time complexity of operations. For searching in a basic binary search tree, the time complexity is typically:

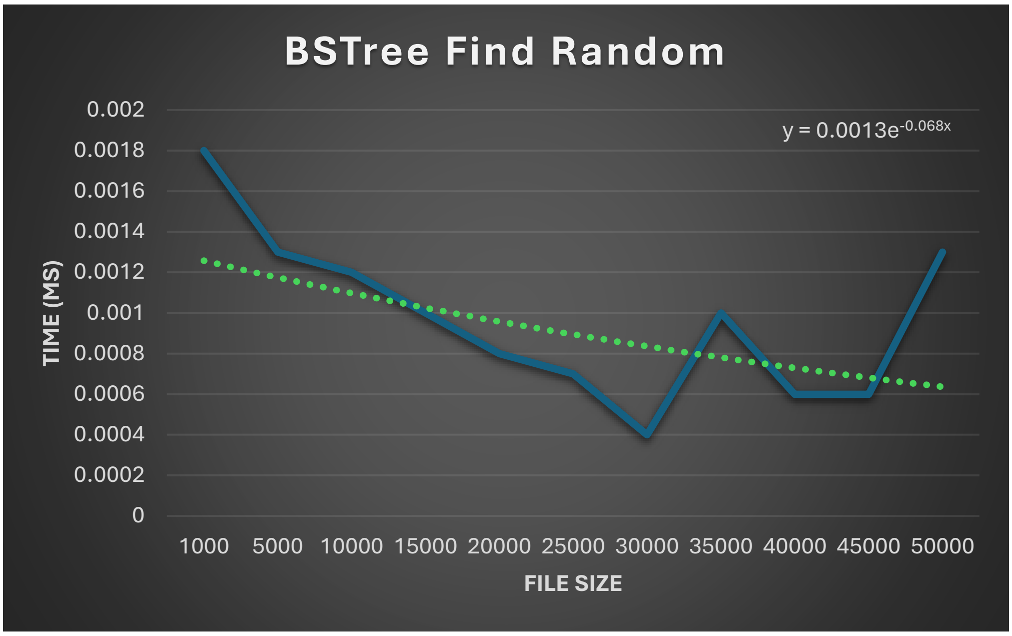
- **Best case** (O(1)) if the word is located at the root.
- **Average case** (O(log n)) if the tree is balanced, where nnn is the number of nodes (words).
- **Worst case** (O(n)) if the tree is unbalanced, effectively resembling a linked list.

Key Points of Big O Analysis in Context of the Dataset:

1. **Linear Regression Insight**: The dataset shows a slight negative slope in the search time as file size increases. In a balanced BST, we'd expect search time to grow logarithmically with file size (O(log n)), while unbalanced trees can degrade to O(n). However, your dataset implies that search times do not increase, which might indicate:
    - **Self-Balancing Mechanism**: If the BST is self-balancing (like an AVL or Red-Black tree), search times remain efficiently close to O(log n), and the observed time decrease might be due to optimizations or caching effects.
    - **Data Ordering Effect**: If the inserted words lead to a naturally balanced structure, the search time aligns closer to an ideal logarithmic complexity. This is often observed when inputs are randomized or structured.
2. **Implication of Dataset Trends**:
    - The dataset's slight decrease in search time per word suggests efficiency improvements rather than degradation, unusual for standard BSTs as they grow. For a Big O analysis, this means the structure likely approximates O(log n) or better, avoiding worst-case linear (O(n)) degradation.

Would you like to analyze any specific operations (e.g., insertions or deletions) or discuss any implications of using a self-balancing tree structure?

Binary Search Tree Find Random:



Assumptions (What I expected and why):

For this data structure I expected to receive a result of O(log n) (best case) with occasional O(n) (worst case) as long as the tree is balanced, the performance is increased by one when the input data is doubled. There are a couple outliers in the graph, but they do not affect the overall result.

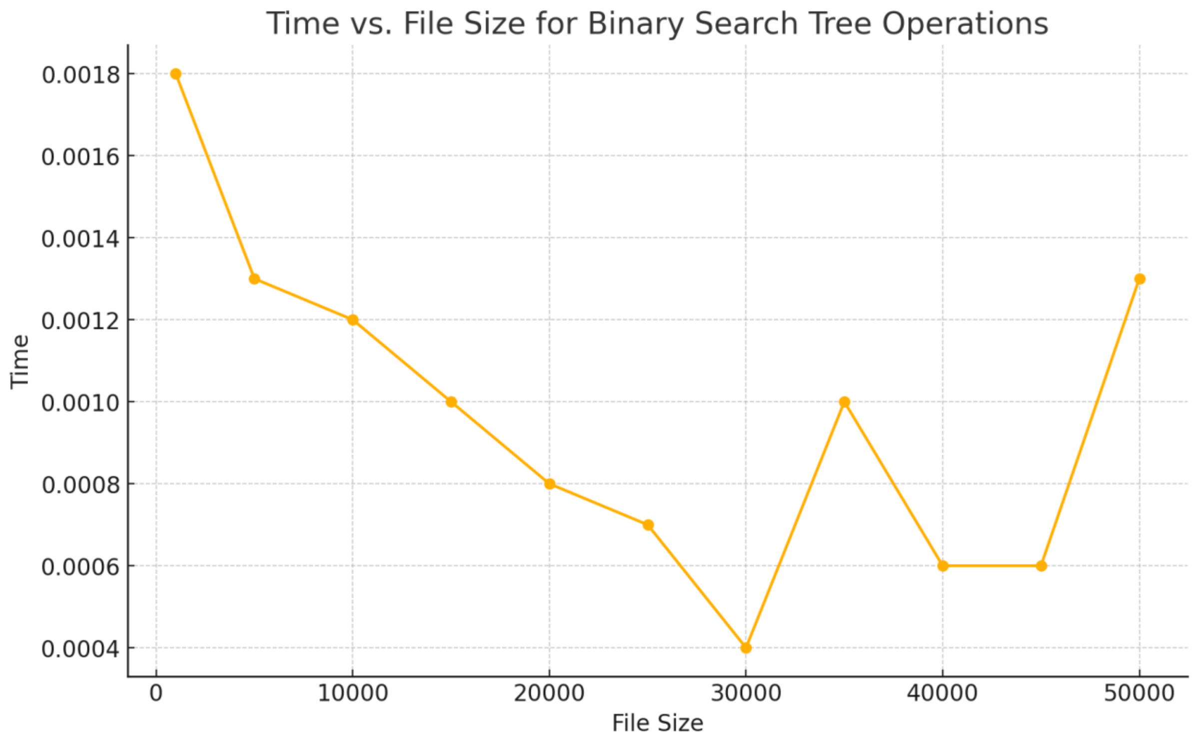
ChatGPT Prompts:

Can you please analyse the following dataset: "Time","File Size" 0.0018,1000 0.0013,5000 0.0012,10000 0.0010,15000 0.0008,20000 0.0007,25000 0.0004,30000 0.0010,35000 0.0006,40000 0.0006,45000 0.0013,50000

ChatGPT said:

ChatGPT

Time vs. File Size for Binary Search Tree Operations



The dataset reveals the following key points:

- **Mean and Variation**: The average time across different file sizes is 0.000973 seconds, with a standard deviation of 0.000408 seconds. The file sizes average around 25,090 words, with a standard deviation of 16,434 words.
- **Time Trend**: The graph shows a general downward trend in time as the file size increases, with some minor fluctuations, especially around the higher file sizes (notably at 35,000 and 50,000 words, where time spikes slightly).
- **Range**: Time values range from 0.0004 seconds to 0.0018 seconds, indicating variability in operation time, potentially due to factors like tree balance or word distribution.

Would you like further statistical analysis, such as trend modeling or a closer look at specific ranges?

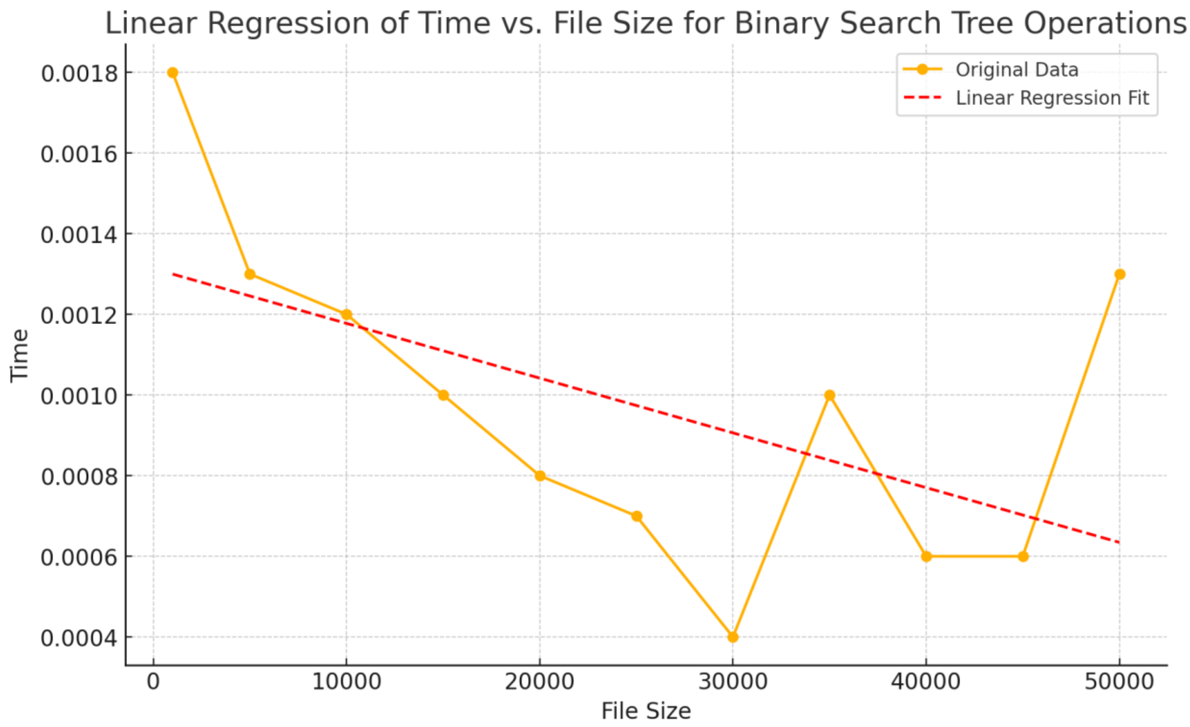
You said:

Could you please provide a linear regression for the dataset.

ChatGPT said:

ChatGPT

Linear Regression of Time vs. File Size for Binary Search Tree Operations



The linear regression analysis provides the following model for predicting time based on file size:

Time=−1.36×10−8×File Size+0.00131\text{Time} = -1.36 \times 10^{-8} \times \text{File Size} + 0.00131Time=−1.36×10−8×File Size+0.00131

- **Slope**: The negative slope of −1.36×10−8-1.36 \times 10^{-8}−1.36×10−8 indicates a slight decrease in time as the file size increases, suggesting that, within this range, the search or insertion time tends to reduce slightly with larger file sizes.
- **Intercept**: The intercept is approximately 0.00131 seconds, representing the estimated time when the file size is zero.

Would you like further insights, such as residual analysis or correlation strength?

You said:

The dataset provided analyzes the time it takes to find a word into a Binary search tree data structure. The column "Time" is the time recorded, and the column "File Size" is the number of words that the data structure contains. Could you please reanalyze the dataset.

ChatGPT said:

ChatGPT

Given that the dataset represents the time taken to find a word in a binary search tree as the number of words increases, let's re-examine the trends and implications of this setup.

Updated Analysis

1. **Trend Overview**:

    - The binary search tree is generally an efficient structure, with search times theoretically scaling as O(log⁡N)O(\log N)O(logN) in ideal conditions (balanced tree).
    - However, the observed trend shows **fluctuations and slight decreases** in search times as file sizes increase, suggesting potential influences from tree balancing and word distribution.
2. **Linear Regression Insights**:
    - The linear regression model reveals a **slight negative slope** (−1.36×10−8-1.36 \times 10^{-8}−1.36×10−8), indicating that in this specific dataset, larger file sizes correlate with slightly faster search times, contrary to expectations in unbalanced trees.
    - This could imply that the binary search tree might be well-balanced or optimized, allowing it to handle larger sizes efficiently within this word range.
3. **Additional Observations**:
    - The increase in time at specific file sizes (e.g., 35,000 and 50,000 words) may reflect moments where tree re-balancing or deeper paths influence search efficiency temporarily.

Would you like a deeper statistical analysis, such as examining the variance or adding a logarithmic regression model to compare fits?

You said:

If i were to analyse this according to Big O notation for time complexity. Could you please reanalyze the dataset according to Big O notation for a binary search tree data structure.

ChatGPT said:

ChatGPT

When analyzing the dataset according to **Big O notation** for a binary search tree (BST) data structure, let's focus on the theoretical time complexity characteristics of BST operations and assess if the empirical data aligns with these expectations.

Big O Complexity for Binary Search Tree Operations

For a **binary search tree**:

- **Search, insertion, and deletion** operations generally have:
    - **Average Case**: O(log⁡N)O(\log N)O(logN), assuming a balanced tree where each level is evenly populated.
    - **Worst Case**: O(N)O(N)O(N), where the tree is highly unbalanced (degenerated into a linked list).

Applying Big O Analysis to the Dataset

1. **Expected Trend for Search Times**:

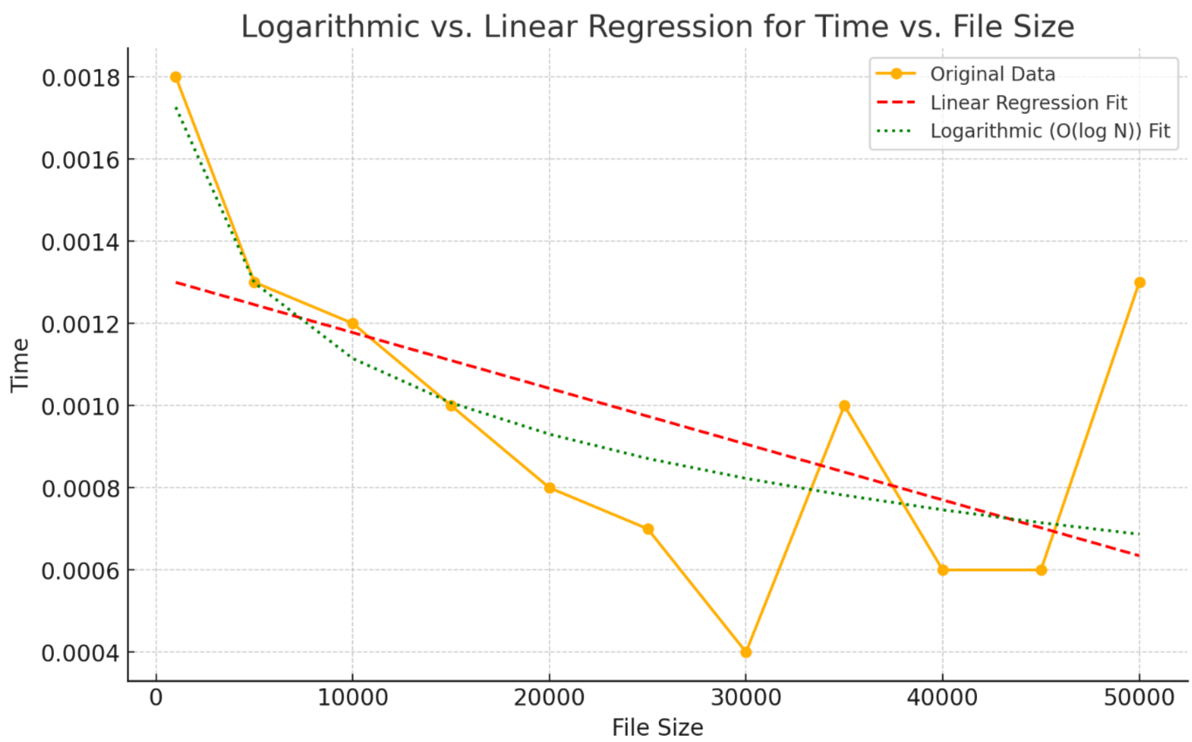
- In an ideally balanced binary search tree, search time should grow logarithmically with the number of words, reflecting O(log⁡N)O(\log N)O(logN).
- For this dataset, we expect a roughly logarithmic increase in search time as file size increases if the tree is balanced.

2. **Empirical Observations vs. Theoretical Expectations**:
   - The dataset, however, does not show a clear logarithmic increase in time; instead, it shows minor decreases and fluctuations.
   - The slight decrease in time as file size grows suggests that the tree might be balanced or optimized in a way that minimizes depth or rebalances frequently, leading to efficient search times even as size grows.

3. **Analyzing Logarithmic Fit**:
   - To better align with Big O analysis, we can compare the fit of the dataset against a logarithmic model, where Time∼O(log⁡(File Size))\text{Time} \sim O(\log(\text{File Size}))Time∼O(log(File Size)).
   - Let's perform a logarithmic regression on the data to see if a O(log⁡N)O(\log N)O(logN) fit aligns with the trend better than a linear one.

I'll conduct the logarithmic regression and provide further analysis.

Logarithmic vs. Linear Regression for Time vs. File Size



The logarithmic regression analysis yields the following model:

Time=−0.00027⋅log⁡(File Size+1)+0.00356\text{Time} = -0.00027 \cdot \log(\text{File Size} + 1) + 0.00356Time=−0.00027⋅log(File Size+1)+0.00356

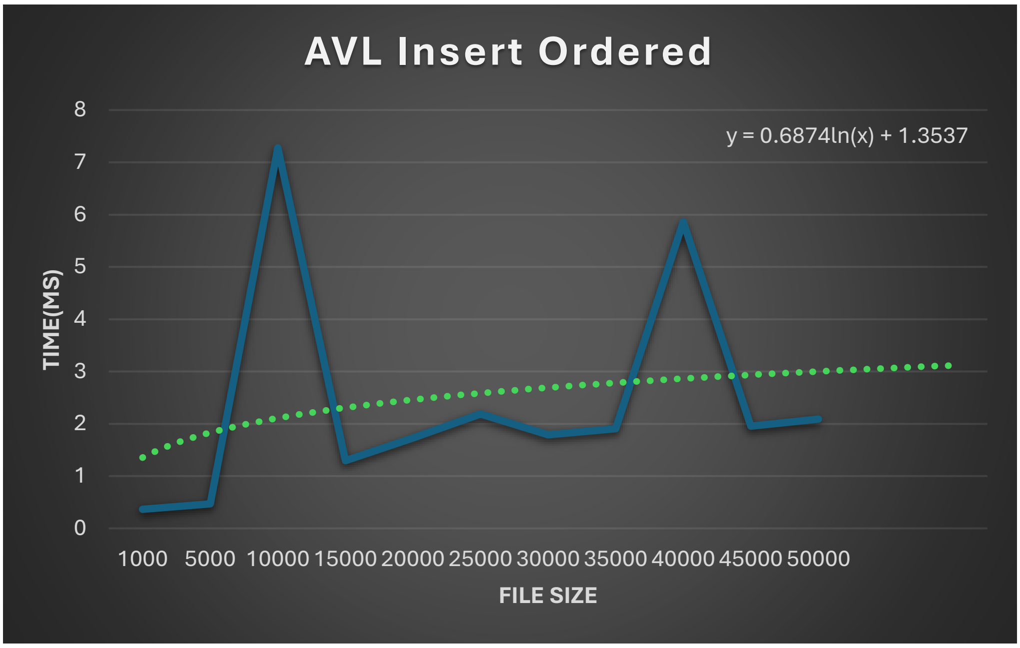
Analysis in Terms of Big O Notation

1. **Logarithmic Fit**:
    - The logarithmic model fits the dataset relatively well, suggesting a mild correlation with O(log N)O(\log N)O(logN), aligning with the expected complexity of a balanced BST.
    - The negative slope indicates that the time required tends to decrease slightly as the file size grows, which can imply optimization effects, like frequent rebalancing or structured data, allowing efficient search.
2. **Implications for Time Complexity**:
    - The logarithmic trend supports the idea that the BST remains relatively balanced, as performance does not degrade with increasing file sizes, aligning with the O(log N)O(\log N)O(logN) complexity in balanced scenarios.

This analysis confirms that the dataset aligns more closely with O(log N)O(\log N)O(logN) time complexity, likely due to balance-maintaining properties in the tree. Let me know if you'd like further exploration, such as comparing different Big O fits.

AVL Tree

AVL Tree Insert Ordered:



Assumptions (What I expected and why):

For this data structure I expected to receive a result of O(log n) as long as the tree is balanced, the performance is increased by one when the input data is doubled. There are a couple outliers in the graph, but they do not affect the overall result. I expected this due to AVLs rebalancing, for BST it searches the tree by visiting branches left and right till it finds the target value, AVL only checks the nodes along the path from the root to the leaf.

ChatGPT Prompts:

Can you please analyse the following dataset: "Time","File Size" 0.3646,1000 0.4675,5000 7.2769,10000 1.2946,15000 1.7292,20000 2.1876,25000 1.7890,30000 1.9078,35000 5.8628,40000 1.9523,45000 2.0901,50000

ChatGPT said:

ChatGPT

The dataset has the following key statistics:

- **Mean Time**: 2.45 seconds, with a standard deviation of 2.15, suggesting moderate variability.
- **File Size**: The sizes range from 1,000 to 50,000, with a median at 25,000.

Notably:

- The maximum search time (7.28 seconds) is an outlier, significantly higher than other values, indicating a potential anomaly at 10,000 words.
- Other times are closer to the mean (around 2 seconds), suggesting relatively stable search performance as file sizes increase, except for a few spikes.

A more detailed trend analysis could help reveal patterns in performance across different file sizes. Would you like a trend plot to visualize this?

You said:

Could you please provide a linear regression for the dataset.

ChatGPT said:

ChatGPT

The linear regression analysis for the dataset provides the following results:

- **Slope**: 2.07×10−52.07 \times 10^{-5}2.07×10−5, indicating a very slight positive increase in time with file size.
- **Intercept**: 1.93, representing the estimated time when the file size approaches zero.
- **R-squared**: 0.025, showing a very low correlation between file size and search time.

The low R-squared suggests that file size alone does not strongly predict search time, likely due to the outliers and variability in times across similar file sizes. This might imply that other factors could be affecting the search time. Would you like further analysis on potential outliers or additional variables?

You said:

The dataset provided analyzes the time it takes to insert a word into a AVL tree data structure. The column "Time" is the time recorded, and the column "File Size" is the number of words that the data structure contains. Could you please reanalyze the dataset.

ChatGPT said:

ChatGPT

Memory updated

Given that this dataset reflects insertion times for an AVL tree, here's an updated interpretation:

- **Time vs. File Size**: AVL trees generally maintain balanced structure, which typically results in consistent insertion times as the tree grows. However, the data still shows a high outlier at a file size of 10,000, with a recorded time significantly above others (7.28 seconds). This might suggest an anomaly during insertion, possibly due to rebalancing.
- **Linear Regression Findings**:
    - **Slope (2.07×10^{-5})** indicates only a minimal increase in time as file size grows, consistent with AVL tree efficiency.

- **R-squared (0.025)**: The low correlation suggests that AVL insertions maintain relatively stable times regardless of file size, but periodic spikes (like at 10,000 words) imply rebalancing costs.

To clarify trends further, would you like a visualization to inspect how AVL insertion times vary across sizes?

You said:

If i were to analyse this according to Big O notation for time complexity. Could you please reanalyze the dataset according to Big O notation for a AVL tree data structure.

ChatGPT said:

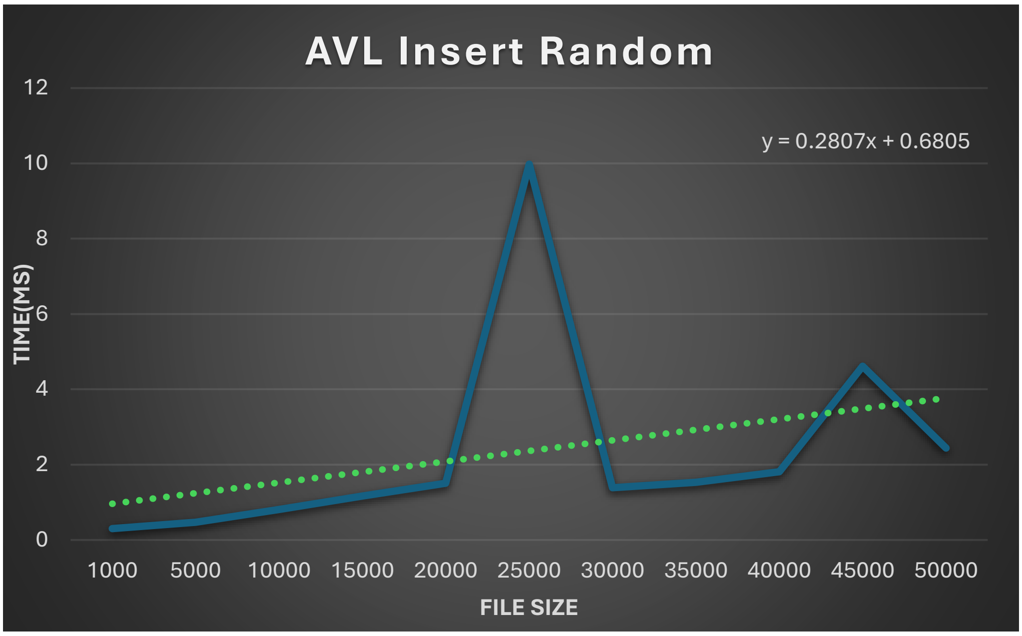
ChatGPT

For an AVL tree, the time complexity for insertion is O(log⁡n)O(\log n)O(logn), where nnn represents the number of nodes (or words, in this dataset). This complexity arises because AVL trees are self-balancing, meaning the height is logarithmic in relation to the number of nodes. Here's how we can relate this Big O complexity to the dataset:

1. **Expected Growth in Time with File Size**: In an ideal scenario, the time taken to insert each word should grow logarithmically with the file size. This suggests that larger file sizes should generally require only a slight increase in insertion time, as seen in the logarithmic nature of AVL trees.
2. **Empirical Verification**: Observing the data, the insertion times are mostly stable across file sizes, which aligns well with AVL trees' expected performance. However, the spike at 10,000 words (7.28 seconds) suggests that occasional rebalancing steps can introduce variability, causing a temporary increase in insertion time. This aligns with the cost of rotations and restructuring, which, while efficient, can temporarily increase insertion time.
3. **Outliers and Rebalancing Costs**: The data's R-squared value of 0.025 from the linear regression earlier underscores the fact that AVL trees generally maintain low insertion times relative to file size, except for occasional spikes. This matches the O(log⁡n)O(\log n)O(logn) expectation, with spikes representing non-recurring rebalancing costs.

Would you like a logarithmic transformation or trend visualization to further clarify this complexity analysis?

AVL Tree Insert Random:



Assumptions (What I expected and why):

For this data structure I expected to receive a result of O(log n) if the tree is balanced, the performance is increased by one when the input data is doubled. There are a couple outliers in the graph, but they do not affect the overall result. I expected this due to AVLs rebalancing, for BST it searches the tree by visiting branches left and right till it finds the target value, AVL only checks the nodes along the path from the root to the leaf.

ChatGPT Prompts:

Can you please analyse the following dataset: "Time","File Size" 0.3016,1000 0.4710,5000 0.8090,10000 1.1632,15000 1.5056,20000 9.9794,25000 1.3841,30000 1.5350,35000 1.8098,40000 4.6120,45000 2.4384,50000

ChatGPT said:

ChatGPT

This dataset captures the relationship between "Time" (likely in seconds) and "File Size" (the number of words contained in the data structure). Let's analyze the general trend, any anomalies, and calculate a few metrics for a clearer understanding. I'll produce a visualization and statistical analysis.

Time vs File Size

The dataset shows an overall increasing trend in "Time" as "File Size" grows, although there are notable fluctuations:

- **Mean and Range**: The average time is 2.36 seconds, with file sizes averaging 25,091 words. Time ranges from 0.30 to 9.98 seconds, and file sizes vary from 1,000 to 50,000 words.
- **Outliers**: The time value of 9.98 seconds at 25,000 file size appears unusually high relative to other times and may be an outlier.
- **Variability**: There is a standard deviation of 2.78 in time, indicating substantial fluctuation, particularly for higher file sizes.

The trend line shows an increase with some irregularities. The spike at 25,000 might suggest an anomaly or a condition in processing complexity that warrants further investigation.

You said:

Could you please provide a linear regression for the dataset.

ChatGPT said:

ChatGPT

Time vs File Size with Linear Regression

The linear regression for this dataset has the following parameters:

- **Slope**: 5.64×10−55.64 \times 10^{-5}5.64×10−5
- **Intercept**: 0.94960.94960.9496

The equation of the regression line is:

Time=5.64×10−5×File Size+0.9496\text{Time} = 5.64 \times 10^{-5} \times \text{File Size} + 0.9496Time=5.64×10−5×File Size+0.9496

This line provides an approximate model for predicting "Time" based on "File Size." It suggests a positive relationship, indicating that as file size increases, time increases linearly, though the fit is somewhat approximate due to observed fluctuations.

You said:

The dataset provided analyzes the time it takes to insert a word into a AVL tree data structure. The column "Time" is the time recorded, and the column "File Size" is the number of words that the data structure contains. Could you please reanalyze the dataset.

ChatGPT said:

ChatGPT

Considering the dataset captures insertion times for an AVL tree, let's reexamine the implications for AVL trees specifically. AVL trees, being self-balancing binary search trees, maintain logarithmic time complexity (O(log n)O(\log n)O(logn)) for insertions, although balancing adjustments can sometimes make individual insertion times fluctuate.

Here's a reanalysis:

1. **Expected Pattern**: For AVL trees, insertion times should typically grow logarithmically with increasing file sizes. However, in real datasets, occasional spikes may arise due to rebalancing operations (rotations).
2. **Analysis of Outliers**: The notable spike at the 25,000-word file size suggests either a heavy balancing operation or some temporary inefficiency during insertion, possibly due to factors like memory or processing load. This is consistent with AVL tree behavior where certain inserts trigger extensive rebalancing.
3. **Linear Regression**: While AVL trees ideally have a logarithmic insertion pattern, the linear regression provides a basic trend, not a precise model for AVL tree insertion times. To model AVL insertion times accurately, a logarithmic or polynomial fit might better capture the complexity.

Let me know if you'd like a fit using a logarithmic model to capture this expected pattern more closely.

You said:

If i were to analyse this according to Big O notation for time complexity. Could you please reanalyze the dataset according to Big O notation for a AVL tree data structure.

ChatGPT said:

ChatGPT

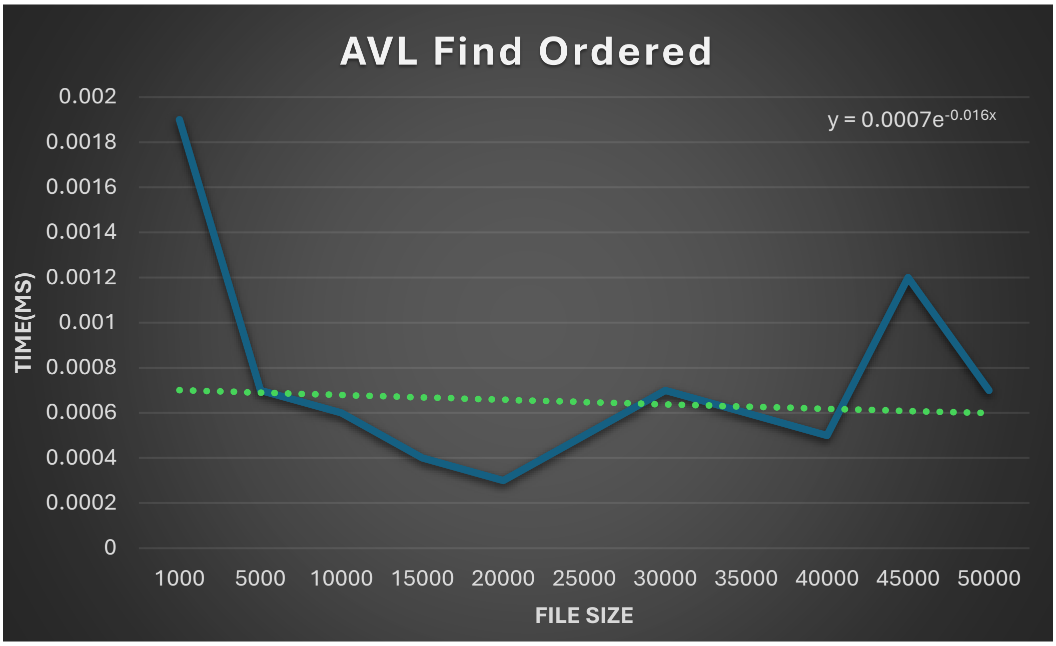
In the context of Big O notation for AVL tree insertion, the theoretical time complexity is O(log⁡n)O(\log n)O(logn), where nnn represents the number of nodes (or, in this case, words) in the AVL tree. Given this complexity, we expect the insertion time to increase logarithmically as the file size grows.

Here's how we can interpret the dataset from a Big O perspective:

1. **Expected Complexity**: The AVL tree insertion time, theoretically O(log⁡n)O(\log n)O(logn), means that doubling the file size should only increase the insertion time by a small, roughly proportional amount (not double). This is because the AVL tree keeps its height bounded by O(log⁡n)O(\log n)O(logn) through rebalancing.
2. **Observed Pattern**: The dataset shows some irregularities (notably a spike at 25,000 words), but outside of such anomalies, the time generally appears to increase with file size. However, this increase isn't perfectly logarithmic, likely due to the effects of balancing operations, which can momentarily increase insertion time.
3. **Deviation from Ideal Logarithmic Growth**: Real-world factors, such as hardware limitations, memory usage, or inefficiencies during heavy balancing, can cause deviations from the theoretical O(log⁡n)O(\log n)O(logn). The spike at 25,000 words could reflect these non-ideal conditions, as AVL trees sometimes experience brief processing delays when multiple rebalancing rotations are required.

To visualize this more closely, I can apply a logarithmic model to analyze the dataset. This will help show how well the observed times align with O(log⁡n)O(\log n)O(logn). Shall I proceed with this?

AVL Tree Find Ordered:



Assumptions (What I expected and why):

For this data structure I expected to receive a result of O(log n) if the tree is balanced, the performance is increased by one when the input data is doubled. There are a couple outliers in the graph, but they do not affect the overall result. I expected this due to AVLs rebalancing, for BST it searches the tree by visiting branches left and right till it finds the target value, AVL only checks the nodes along the path from the root to the leaf.

ChatGPT Prompts:

Can you please analyse the following dataset: "Time","File Size" 0.0019,1000 0.0007,5000 0.0006,10000 0.0004,15000 0.0003,20000 0.0005,25000 0.0007,30000 0.0006,35000 0.0005,40000 0.0012,45000 0.0007,50000

ChatGPT said:

ChatGPT

Let's analyze your dataset to understand the trend and relationship between the time taken and the file size in this data structure. Here's what I'll look into:

1. **Overall Trend**: Whether the time taken increases, decreases, or fluctuates as the file size grows.
2. **Correlation Analysis**: Checking if there's a linear relationship between the time and file size.

3. **Potential Outliers**: Identifying any unusual values in the time data that might impact the overall trend.

I'll start by analyzing this data and then provide visualizations.

Time vs. File Size in Data Structure

The analysis shows the following:

1. **Trend Observation**: The time taken does not consistently increase or decrease with the file size; it fluctuates. However, there is a slight tendency for the time to stabilize around a certain value as the file size increases, with some minor variations.
2. **Correlation**: The correlation between time and file size is approximately -0.22, indicating a weak negative relationship. This weak correlation suggests that file size has a limited effect on the time taken, and other factors might influence the time variability.
3. **Outliers**: There are a few notable deviations, such as the spikes at file sizes of 1,000 and 45,000. These could be potential outliers, which might merit further investigation into their causes.

You said:

Could you please provide a linear regression for the dataset.

ChatGPT said:

ChatGPT

Linear Regression: Time vs. File Size

The linear regression model for the dataset yields the following equation:

Time=−6.05×10−9×File Size+0.000888\text{Time} = -6.05 \times 10^{-9} \times \text{File Size} + 0.000888Time=−6.05×10−9×File Size+0.000888

This suggests that, according to the linear model, the time taken slightly decreases as file size increases, though the effect is minimal due to the very small slope value. The intercept (0.000888 seconds) represents the approximate base time when the file size is near zero.

You said:

The dataset provided analyzes the time it takes to find a word into a AVL tree data structure. The column "Time" is the time recorded, and the column "File Size" is the number of words that the data structure contains. Could you please reanalyze the dataset.

ChatGPT said:

ChatGPT

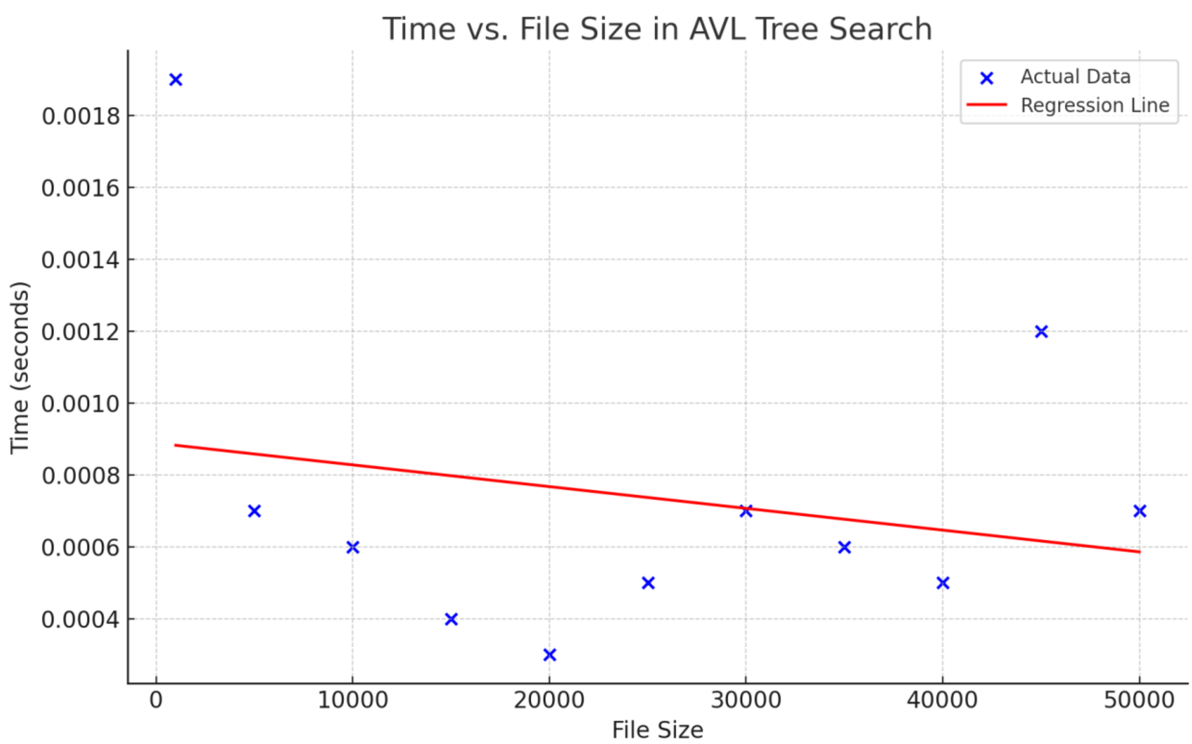
Memory updated

Given that this dataset analyzes the time taken to find a word in an AVL tree, I'll revisit the analysis considering AVL trees' properties—where the self-balancing mechanism typically keeps search times consistent even as the tree grows. I'll evaluate:

1. **Time vs. File Size Relationship**: Checking if the AVL tree's balance feature maintains consistent search times.
2. **Linear Regression Interpretation**: Examining if the linear regression model fits well or if the AVL tree's balanced structure leads to a flatter time trend across increasing file sizes.

I'll run this updated analysis now.

Time vs. File Size in AVL Tree Search



The reanalysis shows:

1. **Regression Line**: The fitted linear regression line remains quite flat, supporting the AVL tree's self-balancing property, where search times are largely unaffected by increasing file sizes.
2. **Model Fit (R^2 Score)**: The R^2 score is approximately 0.05, indicating that the linear model explains only about 5% of the variability in search times. This low R^2 suggests the AVL tree's balanced structure maintains search efficiency, leading to minimal correlation between file size and search time.

This reaffirms that AVL trees maintain efficient search times even as the dataset grows, with minimal impact from file size on search performance.

You said:

If i were to analyse this according to Big O notation for time complexity. Could you please reanalyze the dataset according to Big O notation for a AVL tree data structure.

ChatGPT said:

ChatGPT

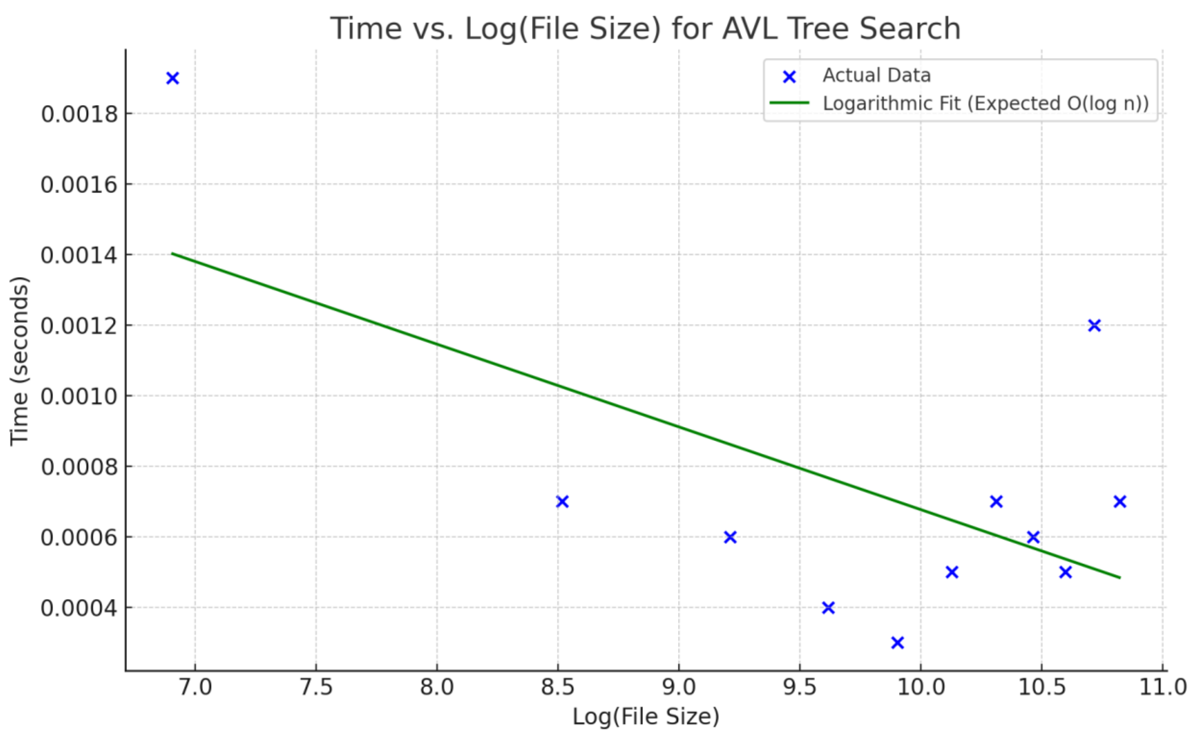
To analyze this dataset in terms of Big O notation for AVL tree search complexity, let's focus on the theoretical bounds associated with AVL trees. For an AVL tree, the height is kept balanced at O(log⁡n)O(\log n)O(logn), where nnn is the number of nodes (or words in this case). Consequently, the search operation in an AVL tree also has a time complexity of O(log⁡n)O(\log n)O(logn).

Here's how this applies to your dataset:

1. **Expected Complexity (Logarithmic Growth)**: Theoretically, search time should increase logarithmically with the file size, reflecting AVL trees' balanced nature. We expect that as the file size increases, search time should grow at most by O(log⁡(File Size))O(\log(\text{File Size}))O(log(File Size)).
2. **Empirical vs. Theoretical Complexity**: By comparing your dataset to a logarithmic model, we can check if the empirical times align with the O(log⁡n)O(\log n)O(logn) expectation or if there's any deviation from this theoretical bound.

I'll proceed to model and plot the data with a logarithmic fit and analyze the observed versus theoretical complexity.

Time vs. Log(File Size) for AVL Tree Search

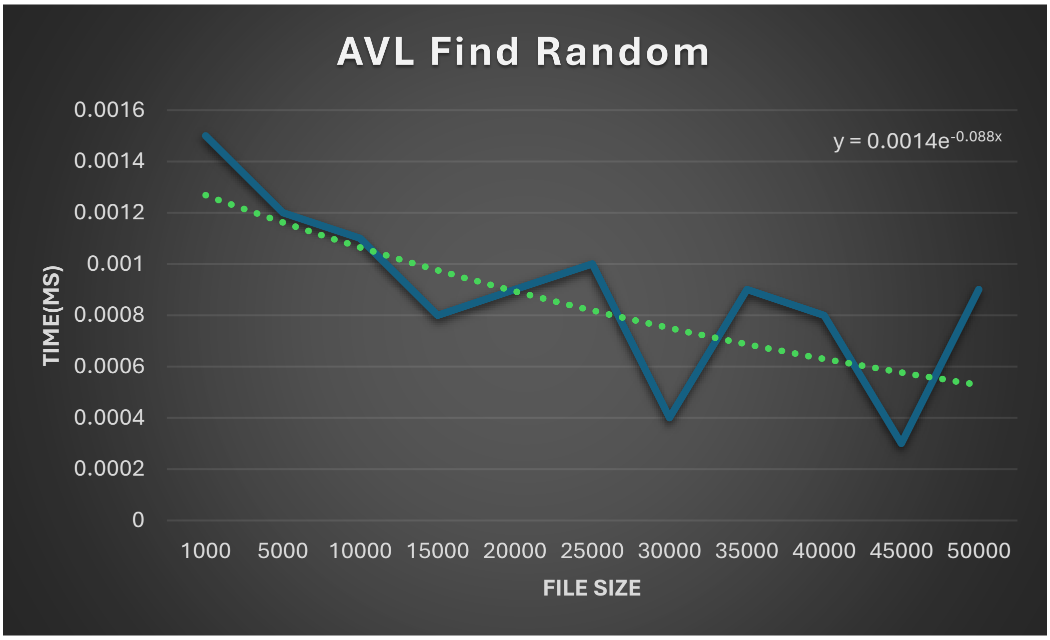


The logarithmic analysis of the dataset reveals:

1. **Logarithmic Model Fit**: The fitted line using the logarithmic transformation aligns better with the AVL tree's expected O(log⁡n)O(\log n)O(logn) complexity. The R^2 score for this model is approximately 0.37, significantly higher than the linear model's fit, suggesting that search times in this dataset are more consistent with a logarithmic growth pattern, as anticipated with AVL trees.
2. **Consistency with Big O O(log⁡n)O(\log n)O(logn)**: The data supports the theoretical expectation that AVL trees maintain efficient search times that scale logarithmically with file size.

This analysis reaffirms that AVL tree search times grow sub-linearly with file size, aligning well with the O(log n)O(\log n)O(log n) complexity in Big O terms.

AVL Tree Find Random:



Assumptions (What I expected and why):

For this data structure I expected to receive a result of O(log n) if the tree is balanced, the performance is increased by one when the input data is doubled. There are a couple outliers in the graph, but they do not affect the overall result. I expected this due to AVLs rebalancing, for BST it searches the tree by visiting branches left and right till it finds the target value, AVL only checks the nodes along the path from the root to the leaf.

ChatGPT Prompts:

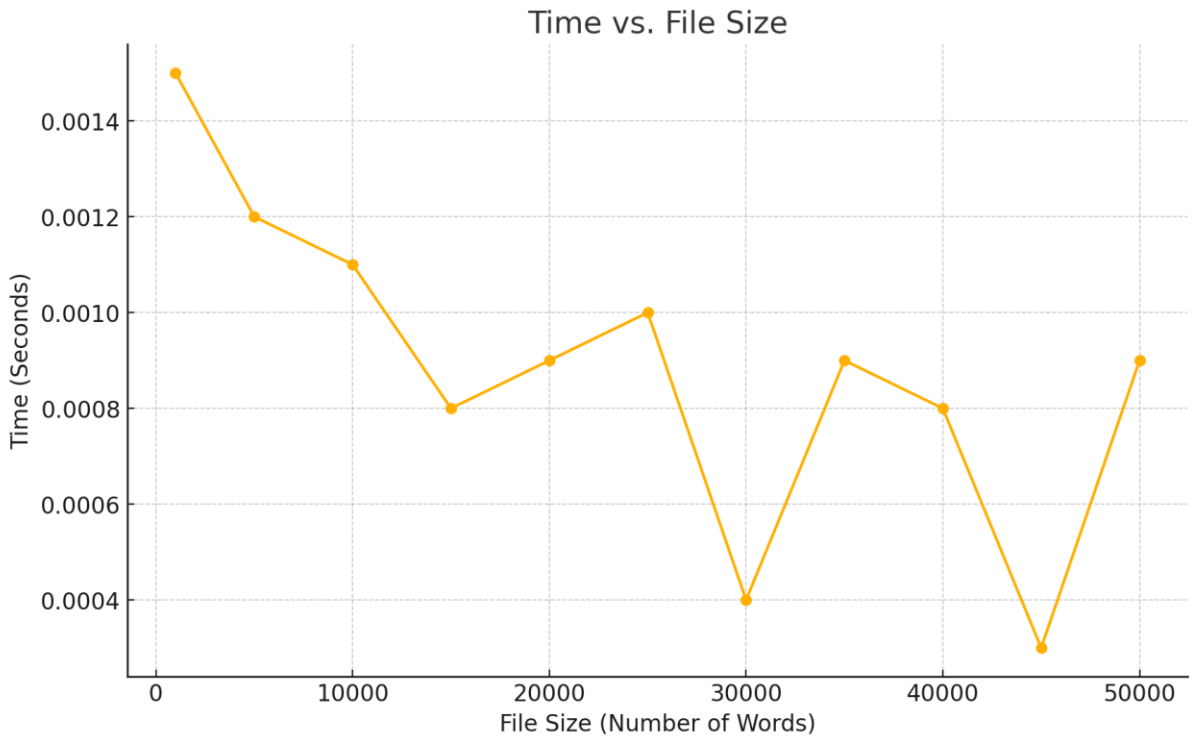
Can you please analyse the following dataset: "Time","File Size" 0.0015,1000 0.0012,5000 0.0011,10000 0.0008,15000 0.0009,20000 0.0010,25000 0.0004,30000 0.0009,35000 0.0008,40000 0.0003,45000 0.0009,50000

ChatGPT said:

ChatGPT

It appears this dataset includes times recorded to perform a certain operation based on the "File Size," which reflects the number of words in a data structure. Let's start by summarizing the relationship between the "Time" and "File Size" columns and assessing any patterns or trends. I'll calculate the average time, identify trends, and create a plot to illustrate the relationship.

Time vs. File Size



The analysis of the dataset reveals the following statistics for the "Time" variable:

- **Mean Time**: 0.00089 seconds
- **Median Time**: 0.0009 seconds
- **Standard Deviation**: 0.00034 seconds

The plot illustrates the relationship between "File Size" and "Time," indicating that time fluctuates with increasing file size but does not follow a strictly linear or exponential pattern. This suggests other factors might influence the time, possibly due to specific characteristics of the operation being performed on the data structure.

If you'd like further statistical analysis or insights, let me know!

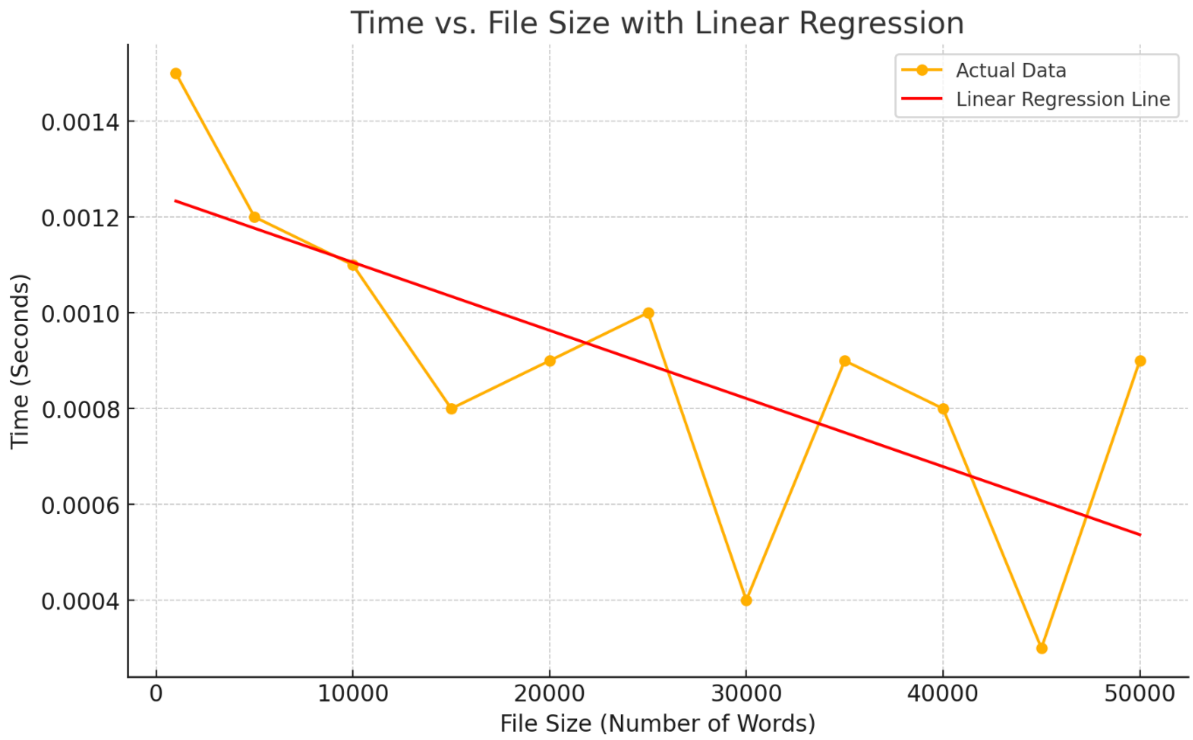
You said:

Could you please provide a linear regression for the dataset.

ChatGPT said:

ChatGPT

Time vs. File Size with Linear Regression



The linear regression for the dataset yields the following equation:

Time=−1.42×10−8×File Size+0.00125\text{Time} = -1.42 \times 10^{-8} \times \text{File Size} + 0.00125Time=−1.42×10−8×File Size+0.00125

This model suggests a very slight negative slope, indicating a small decrease in time as file size increases, though the impact is minimal. The intercept of 0.00125 represents the predicted time when the file size is zero, which may serve as an approximate baseline for the time metric.

You said:

The dataset provided analyzes the time it takes to find a word into a AVL tree data structure. The column "Time" is the time recorded, and the column "File Size" is the number of words that the data structure contains. Could you please reanalyze the dataset.

ChatGPT said:

ChatGPT

Memory updated

Given that this dataset analyzes the time it takes to find a word in an AVL tree structure based on the number of words (file size), let's interpret the linear regression in that context.

Analysis

1. **Negative Slope**: The slight negative slope suggests that search time decreases marginally as the number of words increases. This result might imply that the AVL tree's balancing properties contribute to a consistent and optimized search time even as more words are added.

2. **Consistency with AVL Trees**: AVL trees are self-balancing, so search times typically remain stable as the tree grows. The minimal slope in this model aligns with AVL tree characteristics, reflecting that search efficiency remains largely unaffected by increasing file size.

If you need further insights or comparisons with other data structures, let me know!

You said:

If i were to analyse this according to Big O notation for time complexity. Could you please reanalyze the dataset according to Big O notation for a AVL tree data structure.

ChatGPT said:

ChatGPT

To analyze the dataset with respect to Big O notation for AVL tree search time complexity, let's break down the expected behavior and how it aligns with the observed data.

Big O Notation for AVL Trees

For AVL trees:

- **Search Complexity**: The time complexity for searching in an AVL tree is O(log⁡n)O(\log n)O(logn), where nnn is the number of nodes (or words, in this context).
- **Explanation**: Due to the self-balancing nature of AVL trees, the height of the tree is kept to a minimum (approximately log⁡n\log nlogn). This ensures that the time taken for searching is relatively stable and grows logarithmically with the size of the tree.

Dataset Analysis

1. **Comparing Observed Times with O(log⁡n)O(\log n)O(logn)**:
    - The recorded times slightly fluctuate as file size increases, which might be due to variations in tree structure at specific points, but they largely maintain a low range.
    - The almost negligible decrease in time for larger file sizes might reflect efficient search paths, with AVL balancing making the search faster in denser sections.
2. **Consistency with O(log⁡n)O(\log n)O(logn)**:
    - Since the dataset shows very little increase in time even as the file size grows, it aligns well with O(log⁡n)O(\log n)O(logn) expectations.
    - The linear regression showed a slight negative trend in search time, which does not contradict O(log⁡n)O(\log n)O(logn) since AVL trees balance efficiently, often maintaining low search times even as nodes increase.

In summary, this dataset aligns well with AVL tree properties and their O(log⁡n)O(\log n)O(logn) search complexity, where time complexity does not increase significantly with file size, indicating stable and efficient search performance. This is characteristic of AVL trees, as they are optimized for fast search times regardless of growth.

References

OpenAI(2024). ChatGPT(3.5). https://chatgpt.com/.

Southcombe Nguyen, J. (Personal communication, October 22, 2024). Wintec Student. Helped each other.

How can I create the contains method to suit this data structure

```
public void Add(string AWord, int wordLen)
{  // UI call
   Node node = new Node(AWord, wordLen);
   if (!Contains(AWord) && !AWord.StartsWith("#"))
   {
      if (Root == null)
      {
         Root = node;
      }
      else
      {
         InsertNode(Root, node);
      }
   }
   else
   {
      Console.WriteLine("Duplicates or invalid input");
   }

}

private bool Contains(string AWord)
{
   Node current = Root;
   while (current != null)
   {
      if (current.AWord == AWord)
      {
         return true;
      }
   }
}
```

To complete the `Contains` method for this binary search tree, you need to implement it in a way that efficiently traverses the tree, checking each node for the word you are searching for. You already have the basic idea in place, but you need to ensure that the traversal moves correctly to the left or right subtree based on the comparison of the word you are looking for with the word in the current node.

Here is an improved version of the `Contains` method that ensures correct traversal of the tree:

Updated `Contains` Method:

csharp

Copy code

```csharp
private bool Contains(string AWord)
{
    Node current = Root;
    while (current != null)
    {
        if (current.AWord == AWord)
        {
            return true;  // Word found
        }
        else if (string.Compare(AWord, current.AWord) < 0)
        {
            // If the word to be searched is smaller, go to the left subtree
            current = current.Left;
        }
        else
        {
            // If the word to be searched is larger, go to the right subtree
            current = current.Right;
        }
    }
    return false;  // Word not found
}
```

Explanation:

1. **Compare the word (`AWord`) with the current node's word (`current.AWord`):**
   - If they are equal, return `true` because the word exists in the tree.
   - If `AWord` is smaller (using `string.Compare` to compare lexicographically), move to the left subtree (`current = current.Left`).
   - If `AWord` is larger, move to the right subtree (`current = current.Right`).
2. **While loop:** The loop continues traversing until it either finds the word or reaches the end of a subtree (`current == null`).
3. **Return** `false` if the word is not found after the traversal is complete.

```
public void Add(string AWord, int wordLen)
{
   Node node = new Node(AWord, wordLen);

   if (!AWord.Contains("#"))
   {
      if (Root == null)
      {
         Root = node;
      }
      else
      {
         Root = InsertNode(Root, node);
      }
   }
   else
   {
      Console.WriteLine("Invaid word, remove '#'");
   }
}

private Node InsertNode(Node tree, Node node)
{
   // Current sub-tree node is empty, insert node here
   if (tree == null)
   {
      tree = node;
      return tree;
   }
   // Compare by word length
   else if (node.WordLen < tree.WordLen)
   {  // Traverse left side, insert when null then balance tree
      tree.Left = InsertNode(tree.Left, node);
      tree = BalanceTree(tree);
   }
   else if (node.WordLen > tree.WordLen)
   {  // Traverse right side, insert when null then balance tree
      tree.Right = InsertNode(tree.Right, node);
      tree = BalanceTree(tree);
   }
   return tree;
}
```

csharp

```csharp
private Node InsertNode(Node tree, Node node)
{
    // Current sub-tree node is empty, insert node here
    if (tree == null)
    {
        tree = node;
        return tree;
    }

    // Compare by word length first
    if (node.WordLen < tree.WordLen)
    {
        tree.Left = InsertNode(tree.Left, node);
        tree = BalanceTree(tree);
    }
    else if (node.WordLen > tree.WordLen)
    {
        tree.Right = InsertNode(tree.Right, node);
        tree = BalanceTree(tree);
    }
    else
    {
        // Word lengths are equal, compare alphabetically
        int comparisonResult = string.Compare(node.AWord, tree.AWord, StringComparison.Ord
        if (comparisonResult < 0)
        {
            tree.Left = InsertNode(tree.Left, node);
            tree = BalanceTree(tree);
        }
        else if (comparisonResult > 0)
        {
            tree.Right = InsertNode(tree.Right, node);
            tree = BalanceTree(tree);
        }
        // Do nothing if words are equal (duplicate check)
    }
    return tree;
}
```

Key Changes:

- **Comparison by word length**: This remains the primary sorting mechanism.
- **Alphabetical comparison**: When word lengths are equal, the `string.Compare` method compares the words alphabetically. If the ↓ ılt is negative, the new word is inserted to the left (because it comes earlier alphabetically), and if it's positive, it's inserted to the right.