William Fishburn

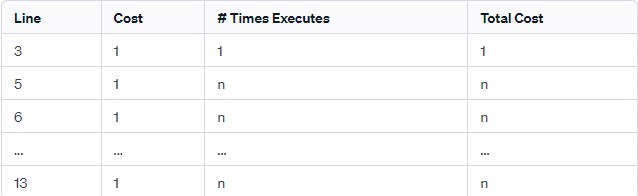
Project One - Runtime Analysis

CS 300 - DSA

**BST**

**Total Cost : 10 + 7n**

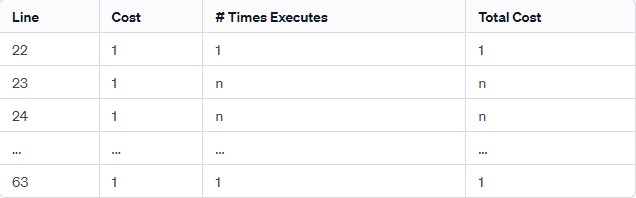
**Worst-case Running Time (Big O): O(n)**



**Hash Table:**

**Total Cost**: 42 + n

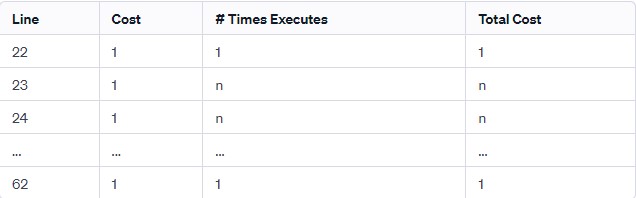
**Worst-case Running Time (Big O):** O(n)



**Vector:**

**Total Cost**: 41 + n

**Worst-case Running Time (Big O)**: O(n)



**Vector:**

**Pros:**

Dynamic Size: Easily accommodates changes in size.

Sequential Access: Efficient for sequential data access.

Simplicity: Simple to implement and use.

**Cons:**

Inefficient Search: Searching has a linear time complexity (O(n)).

**Hash Table:**

**Pros:**

Fast Lookup: Provides constant-time average lookup.

Dynamic Sizing: Adjusts well to changing data sizes.

Effective for Key-Value Pairs: Suited for scenarios with unique identifiers.

**Cons:**

Hash Collisions: Handling collisions can add complexity.

No Inherent Order: Lacks inherent order of elements.

**Binary Search Tree (BST):**

**Pros:**

Ordered Structure: Maintains order for efficient searching, insertion, and deletion.

Balanced Trees Improve Performance: Logarithmic height ensures optimal performance.

In-order Traversal: Facilitates retrieval of elements in sorted order.

**Cons:**

Sensitive to Input Order: Performance may suffer with already sorted input.

Complexity of Operations: Some operations (insertion, deletion, balancing) can be more intricate.

**Recommendation:**

Given the task of loading and sorting bids, a Binary Search Tree (BST) is recommended. The balanced nature of a BST ensures logarithmic time complexity (O(log n)) for search operations, providing efficient performance as the dataset grows. Moreover, in-order traversal allows the printing of bids in ascending order with a time complexity of O(n). While a hash table offers O(1) average case lookup, the ordering requirement leans toward the strengths of a BST. For simplicity and dynamic resizing, a vector might be considered but is less suitable for efficient bid sorting. The final decision should align with the specific operational needs of the application, considering the trade-offs between time complexity, ordering requirements, and ease of implementation.