

## **Data Structure Trie project**

Tries are specialized tree-based data structures optimized for storing and retrieving strings. Unlike binary search trees. Tries organize data by character prefixes, making them particularly effective for operations such as autocomplete. This project implementing a Trie in C++ and evaluates its performance through controlled experiments.

---

The Trie was designed to support the following core operations:

**Insert:** Add words character by character into the Trie.

**Search:** check whether a word exists in the Trie.

**Delete:** Removes words while maintaining structural integrity.

**Autocomplete:** Returns suggestions based on given prefix.

---

### Dataset Generation

10,000 random words of length 5-9 characters were generated using the <random> library

### **Experiment Setup:**

- Inset all 10,000 words into the Trie
- Search for 1,000 random words (half present, half absent)
- Delete 500 random words.
- Measure execution time for each operation using <chrono> in milliseconds.

I used Ubuntu Terminal and compilation with “g++ -std=c++14-wall”

Operation	Trie (10k words)
Insert	8 ms
Search	~0ms (1000 queries)
Delete	~0ms (500 words)

Analysis:

- **Insertion:** The Trie handled 10,000 word insertions in 8 ms, demonstrating efficient scalability for string-heavy datasets.
- **Search:** Search operations completed almost instantly, highlighting the Trie’s strength in prefix-based lookups.
- **Deletion:** Deletion was similarly efficient, with negligible timing impact when removing 500 words.

**Conclusion:**

This project demonstrated that a Trie implemented in C++14 can efficiently manage large sets of words. Insertion of 10,000 randomly generated words completed in only 8 milliseconds, while search and deletion operations were effectively instantaneous.

These results highlight the strength of Tries for prefix-based lookups and confirm their usefulness in applications such as autocomplete and spell checking.

The experiments showed that the data structure scales well for string-heavy tasks and maintains structural integrity during deletions. Overall, the project confirmed that Tries are a practical and reliable choice for text-processing systems where speed and accuracy are essential.