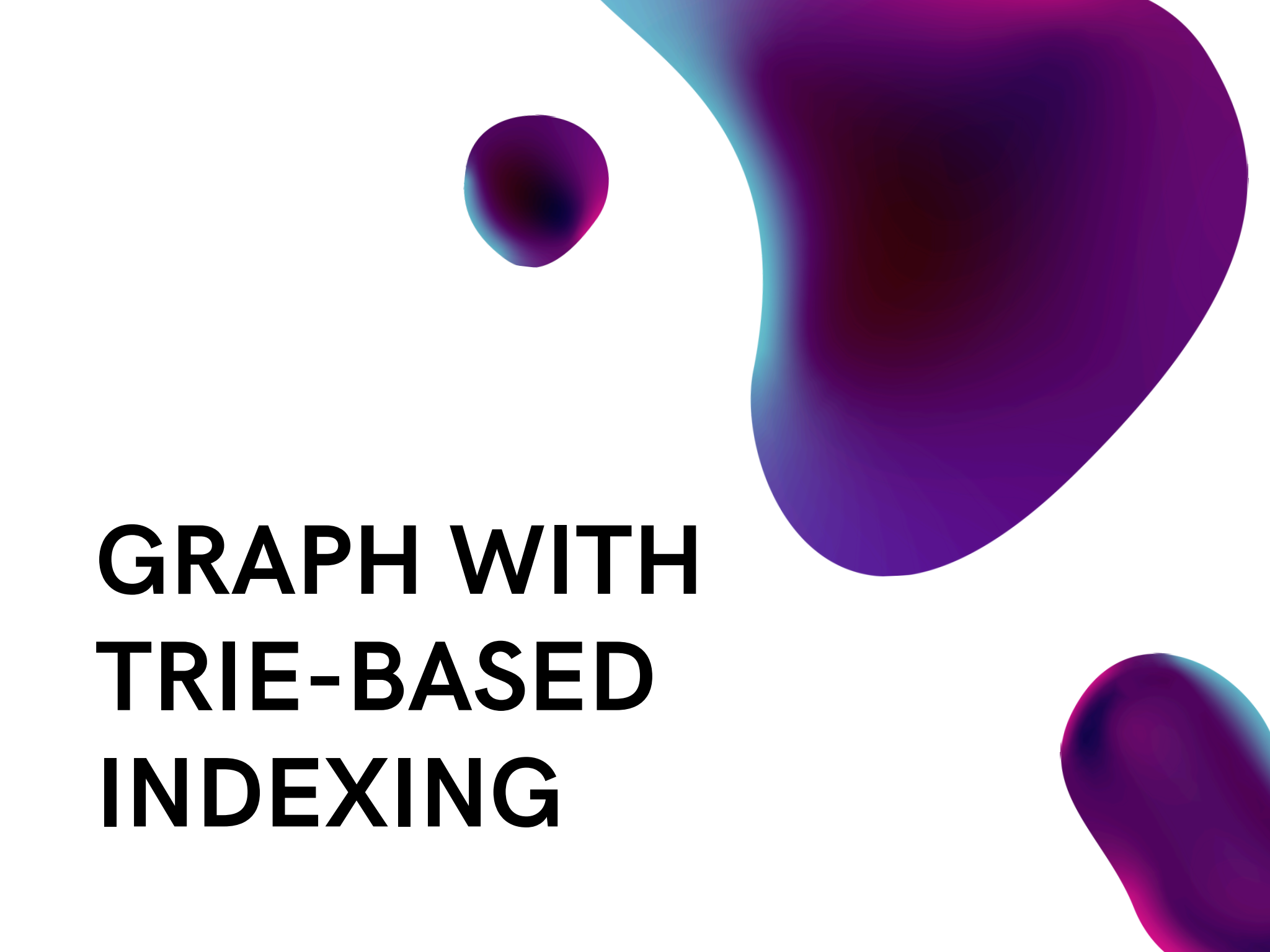
GRAPH WITH
TRIE-BASED
INDEXING

01

This project focuses on designing and implementing a hybrid data structure called "Graph with Trie-Based Indexing."

DATA STRUCTURES

A Graph is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph.

The Trie data structure is a tree-like data structure used for storing a dynamic set of strings. It consists of nodes connected by edges. Each node represents a character or a part of a string.

03

Why Graph and Trie-indexing?

Tries inherently provide a **hierarchical organization** of data based on shared prefixes.
Tries are known for their efficient **prefix-based searching**.
Graphs excel at representing **relationships between entities**.
By combining the strengths of tries and graphs, the hybrid data structure can provide **improved performance** for specific problem domains.

04

Practical Applications

Browser History

Auto-Complete

Spell checkers

Prefix Matching Algorithm

Time Complexity

05

Time complexity

Insertion

For inserting the string we are connecting each letter in the string through for loop. So the time complexity is going to depend on the length of the string, so if the length of the string we are inserting is N then the time complexity of inserting string is O(N).

Searching

For searching the string it is going to depend on length of the string () the time complexity of searching the string is O(N).If we compare the time complexity with the trie which is also O(N) ,so for searching the string our hybrid data structure and in trie takes the same time.

Deletion

For searching the string it is going to depend on length of the string (N) and the total number of vertices(V) int the graph because in the worst case we have to search every vertex int the graph so then the time complexity of searching the string is O(N*V).

07

Conclusion

This project focus on implementation of the hybrid data structure graph with trie-based indexing and we implemented using adjacent list.
Overall, the project demonstrates the practicality, effectiveness, and potential of the hybrid data structure, highlighting its usefulness in real-world scenarios and providing valuable insights for future enhancements.

Thank you

-CB.EN.U4CSE22522
-CB.EN.U4CSE22530
-CB.EN.U4CSE22544
-CB.EN.U4CSE22545
-CB.EN.U4CSE22546