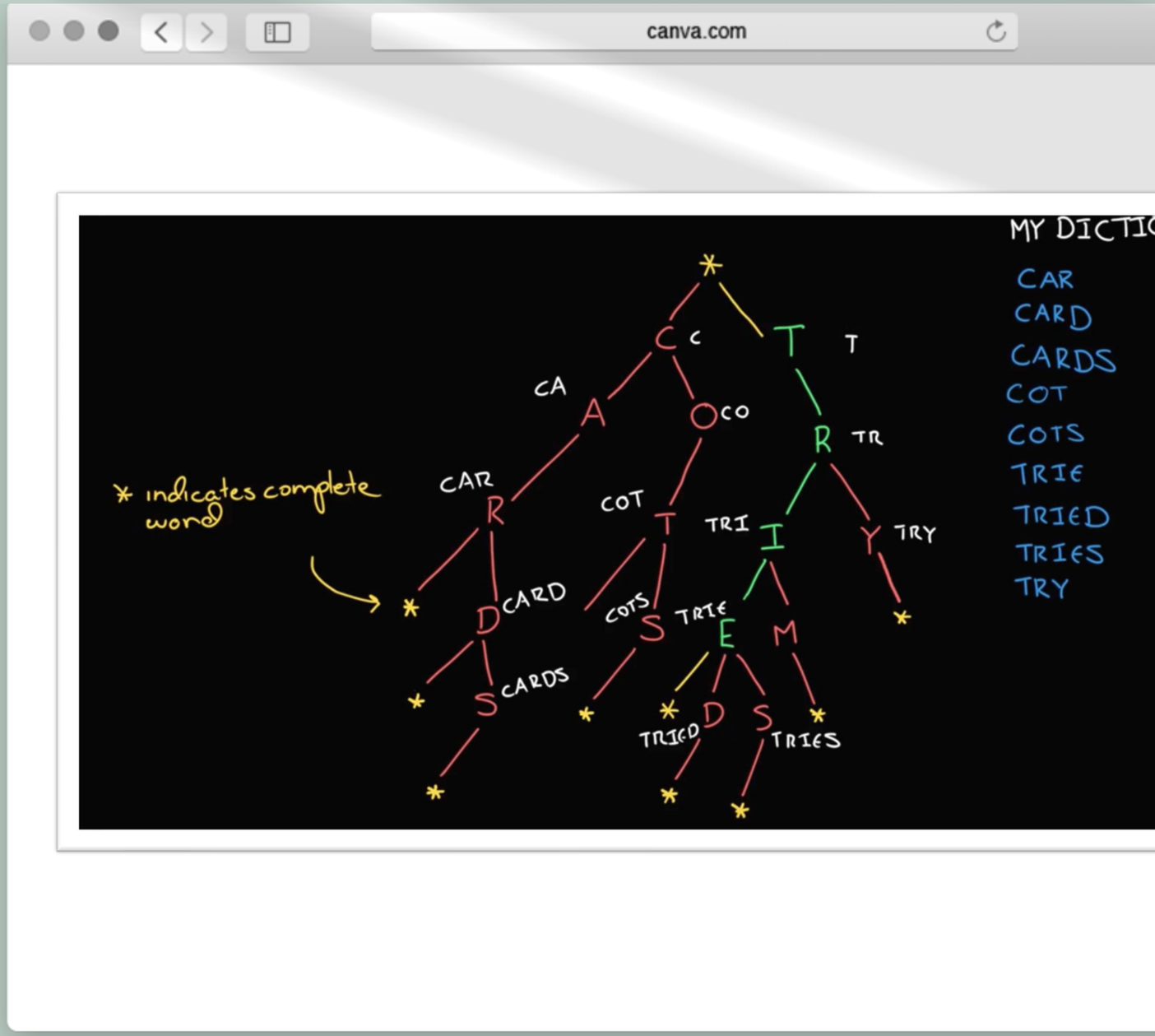
BÁO CÁO ĐỒ ÁN TRIE

12/12/2024

23120100_Trần Minh Trọng

23120100@student.hcmus.edu.vn

GIỚI THIỆU SƠ LƯỢC CẤU TRÚC TRIE



SƠ LƯỢC CẤU TRÚC TRIE

|GIỚI THIỆU:

- **Trie** (đọc là *"try"*) là một cấu trúc dữ liệu cây được sử dụng chủ yếu để lưu trữ và tìm kiếm từ trong từ điển hoặc các chuỗi ký tự.
- Điểm đặc biệt của Trie là các nút con được tổ chức theo từng ký tự của từ. Mỗi đường đi từ gốc đến một nút lá tương ứng với một từ trong từ điển.

|CẤU TRÚC CỦA TRIENODE:

- **children**: Mảng gồm 26 phần tử (ALPHABET_SIZE), mỗi phần tử tương ứng với một chữ cái trong bảng chữ cái (a-z).
- **isEndOfWord**: Biến Boolean dùng để đánh dấu khi một nút là kết thúc của một từ.

CÁC HÀM CÀY ĐẶT CHƯƠNG TRÌNH

1. void insert(TrieNode* root, const string& word)

2. TrieNode* searchPrefix(TrieNode* root, const string& prefix)

3. void suggestWords(TrieNode* root, string currentPrefix, int& t)

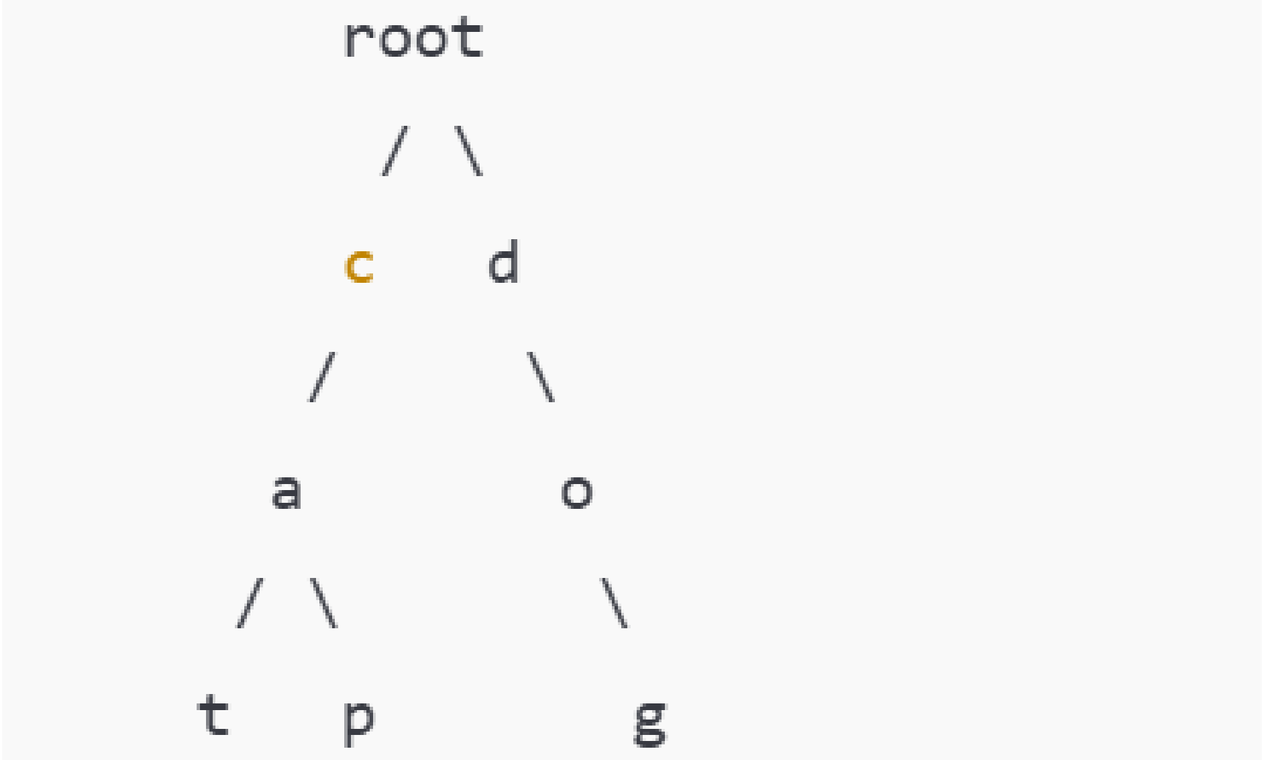
4. void autocomplete(TrieNode* root, const string& prefix)

5. void deleteWord(TrieNode* root, const string& word, bool& isDeleted, int depth = 0)

VÍ DỤ:

Thêm các từ: "cat", "cap", "dog"

Trie sẽ có cấu trúc như sau:



GIỚI THIỆU CÁC THÔNG TIN BỔ SUNG

|DỮ LIỆU SỬ DỤNG:

•**Từ điển**: Chương trình đọc dữ liệu từ tệp words.txt chứa danh sách 19500 từ vựng Tiếng Anh cơ bản.
•**Giới hạn**: Chỉ chấp nhận từ có độ dài từ 3 ký tự trở lên để đảm bảo tính hiệu quả.

|CẢI TIẾN:

- Hàm searchWord() để kiểm tra từ khóa có trong cấu trúc cây chưa.

```
bool searchWord(TrieNode* root, const string& word)
```

SO SÁNH VỚI

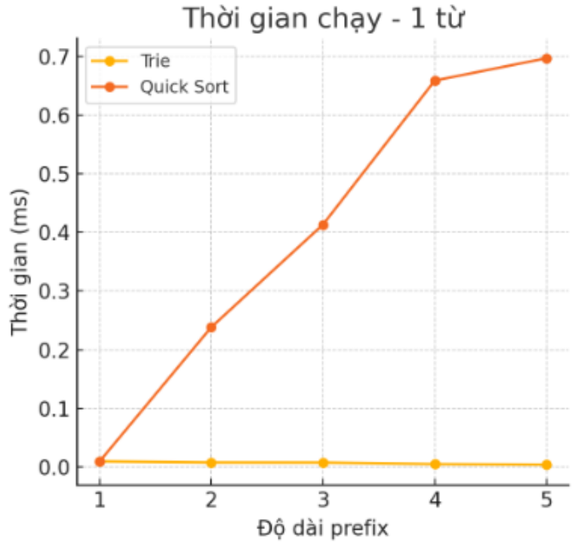
QUICK SORT

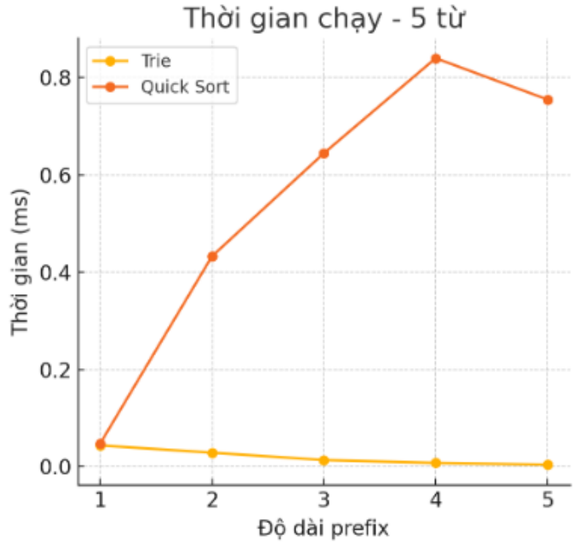
BẢNG THỰC NGHIỆM

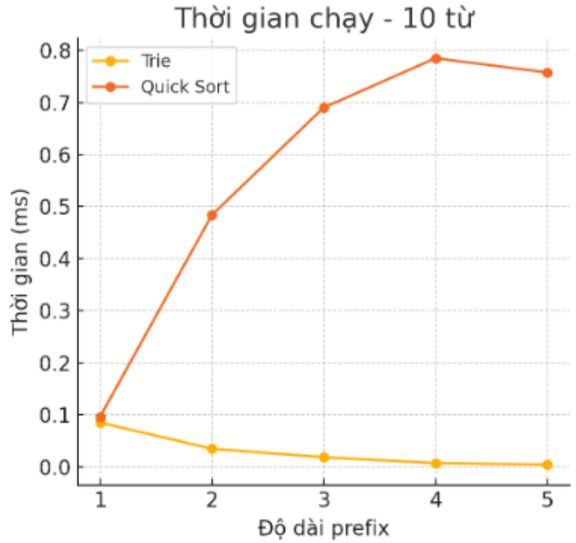
TRIE						
Độ dài chuỗi prefix		1	2	3	4	5
Trung bình thời gian chạy đưa ra (ms)	1 từ	0.009452	0.007527	0.007407	0.004743	0.003801
	5 từ	0.043510	0.028527	0.013391	0.007439	0.003891
	10 từ	0.084810	0.034528	0.018185	0.007023	0.003965
	15 từ	0.142760	0.047350	0.020730	0.006555	0.003988
Trung bình số phép so sánh khi đưa ra	1 từ	15	19	14	12	8
	5 từ	258	179	103	44	25
	10 từ	584	319	145	48	26
	15 từ	897	442	176	50	26

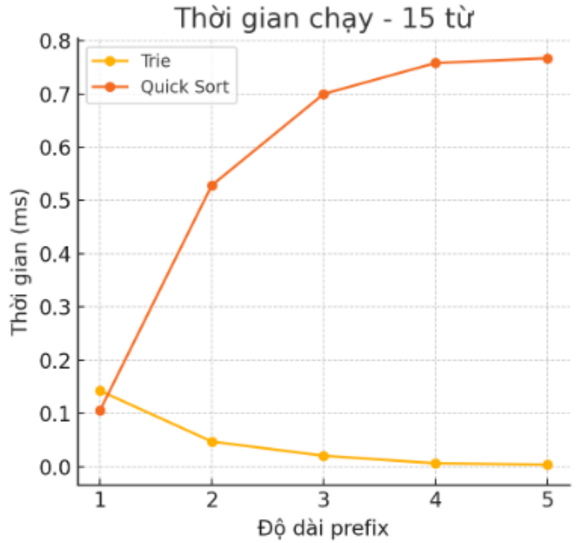
Quick_Sort						
Độ dài chuỗi prefix		1	2	3	4	5
Trung bình thời gian chạy đưa ra (ms)	1 từ	0.008757	0.238018	0.413255	0.658888	0.697041
	5 từ	0.047424	0.432752	0.644606	0.840213	0.755233
	10 từ	0.095871	0.483799	0.690132	0.785196	0.757718
	15 từ	0.105635	0.528360	0.699902	0.757990	0.767077
Trung bình số phép so sánh khi đưa ra	1 từ	137	5784	9408	15236	16897
	5 từ	528	12971	17584	19210	19372
	10 từ	1132	13755	18471	19372	19400
	15 từ	1554	14546	18882	19393	19405

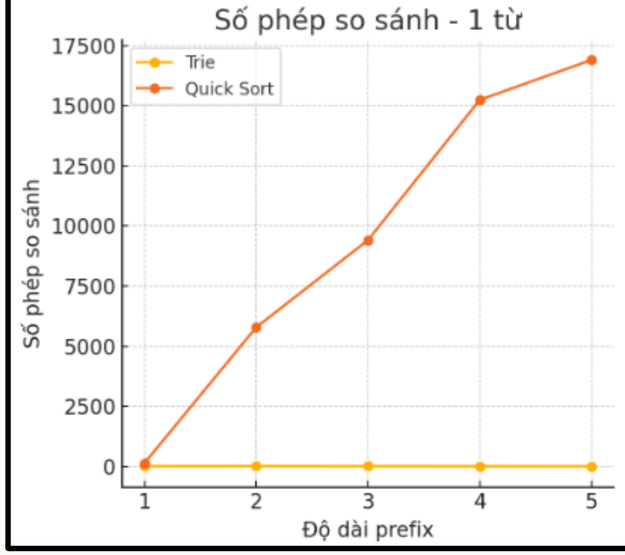
SO SÁNH TRIE VỚI QUICK_SORT

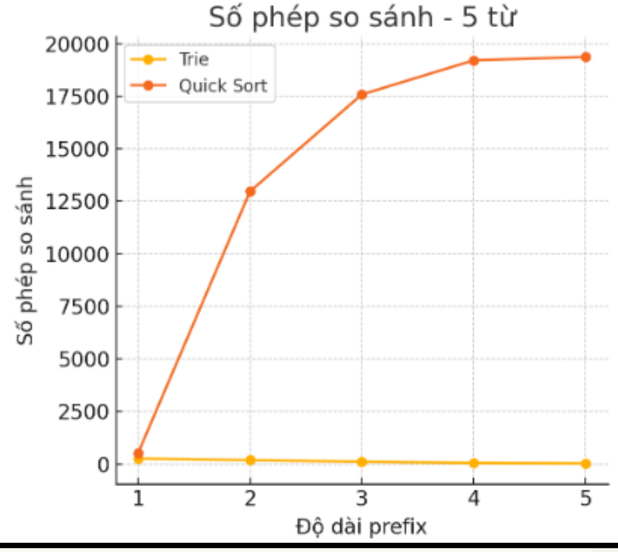


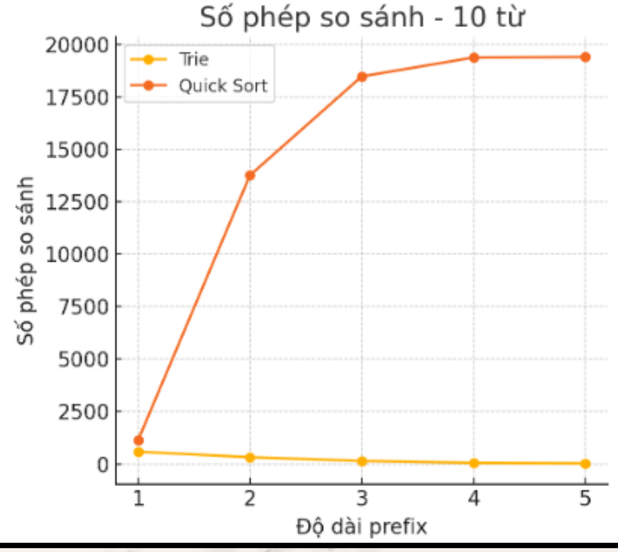


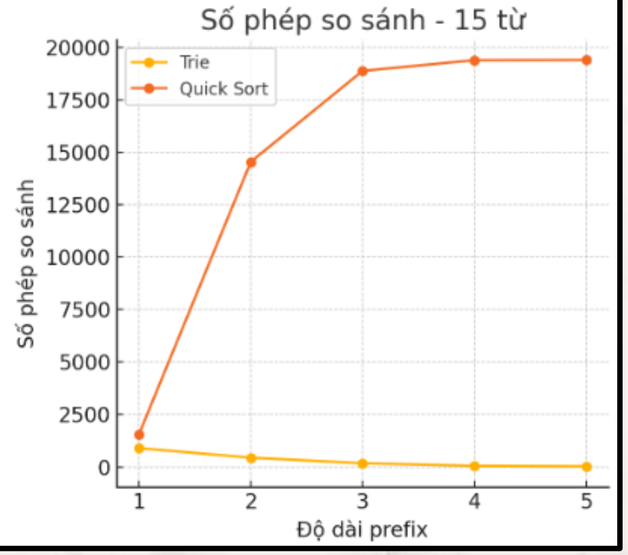












SO SÁNH TRIE VỚI QUICK_SORT

- Trie:

- Độ phức tạp tốt nhất: O(L), với L là độ dài của chuỗi.
- Độ phức tạp trong trường hợp xấu nhất (với dữ liệu bị phân tán và không tối ưu): O(L*alphabet size)
- Trie sử dụng nhiều bộ nhớ hơn vì lưu trữ các nút trung gian và cấu trúc phân cấp.

- Quick Sort:

- Độ phức tạp trung bình: O(nlogn) với n là số phần tử.
- Độ phức tạp trường hợp xấu nhất: O(n^2) nếu pivot chọn không tốt.
- Quick Sort yêu cầu ít bộ nhớ hơn vì không cần cấu trúc phụ như Trie.

SO SÁNH DỰA TRÊN THỰC NGHIỆM

Tiêu chí	Trie	Quicksort
Thời gian chạy trung bình (ms)	Trie hoạt động nhanh hơn ở mọi độ dài prefix, đặc biệt khi prefix càng dài. Nhờ cấu trúc cây, Trie chỉ cần duyệt theo thứ tự các ký tự, giúp giảm thời gian xử lý đáng kể.	Quick Sort chậm hơn so với Trie, đặc biệt với prefix ngắn. Khi prefix ngắn, thuật toán phải xử lý nhiều bước sắp xếp trước khi tìm kết quả.
Số phép so sánh trung bình khi đưa ra	Với prefix ngắn (1-2 ký tự), Trie thực hiện nhiều phép so sánh hơn Quick Sort vì cần duyệt qua nhiều nút con. Tuy nhiên, khi prefix dài hơn, Trie thực hiện ít phép so sánh hơn nhờ rút ngắn số lượng nút cần kiểm tra.	Quick Sort thực hiện số phép so sánh ổn định hơn trên tất cả các độ dài prefix nhờ bản chất của thuật toán sắp xếp. Tuy nhiên, điều này không đồng nghĩa với việc nó nhanh hơn.
Khi nào hiệu quả hơn?	Trie có khả năng tìm kiếm nhanh, đặc biệt với prefix dài, nhờ đặc trưng của cấu trúc cây mà mỗi cấp độ tương ứng với một ký tự trong prefix.	Quick Sort phù hợp hơn khi xử lý lượng dữ liệu nhỏ và khi yêu cầu xử lý tổng quát hơn, không chỉ tập trung vào prefix cụ thể.
Ưu điểm nổi bật	Trie hiệu quả hơn trong các tình huống có dữ liệu lớn và khi cần tìm kiếm với prefix dài. Điều này thường thấy trong các ứng dụng như từ điển hoặc hệ thống tìm kiếm gợi ý tự động.	Quick Sort có cấu trúc đơn giản, dễ triển khai và ổn định trong nhiều trường hợp, giúp đảm bảo tính đúng đắn và dễ bảo trì.
Nhược điểm chính	Trie yêu cầu bộ nhớ cao hơn, vì cần lưu trữ toàn bộ cấu trúc cây, bao gồm cả các nút rỗng để duy trì đường dẫn cho các prefix khác nhau.	Quick Sort không tối ưu khi tìm kiếm với prefix dài, vì số lượng phép so sánh tăng nhanh, làm giảm hiệu suất tổng thể.

VIDEO DEMO


****
CHUONG TRINH TU DIEN VOI TRIE
****
1. Goi y tu khoa
2. Them tu khoa
3. Xoa tu khoa
4. Thoat chuong trinh
5. Kiem tra tu khoa
Chon mot so (1-5):

CÁC TÀI LIỆU THAM KHẢO

1.Cấu trúc dữ liệu và Giải thuật – Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.

2.Trie Data Structure – GeeksforGeeks: https://www.geeksforgeeks.org/trie-insert-and-search/

3.Data Structures and Algorithms in C++ – Adam Drozdek.