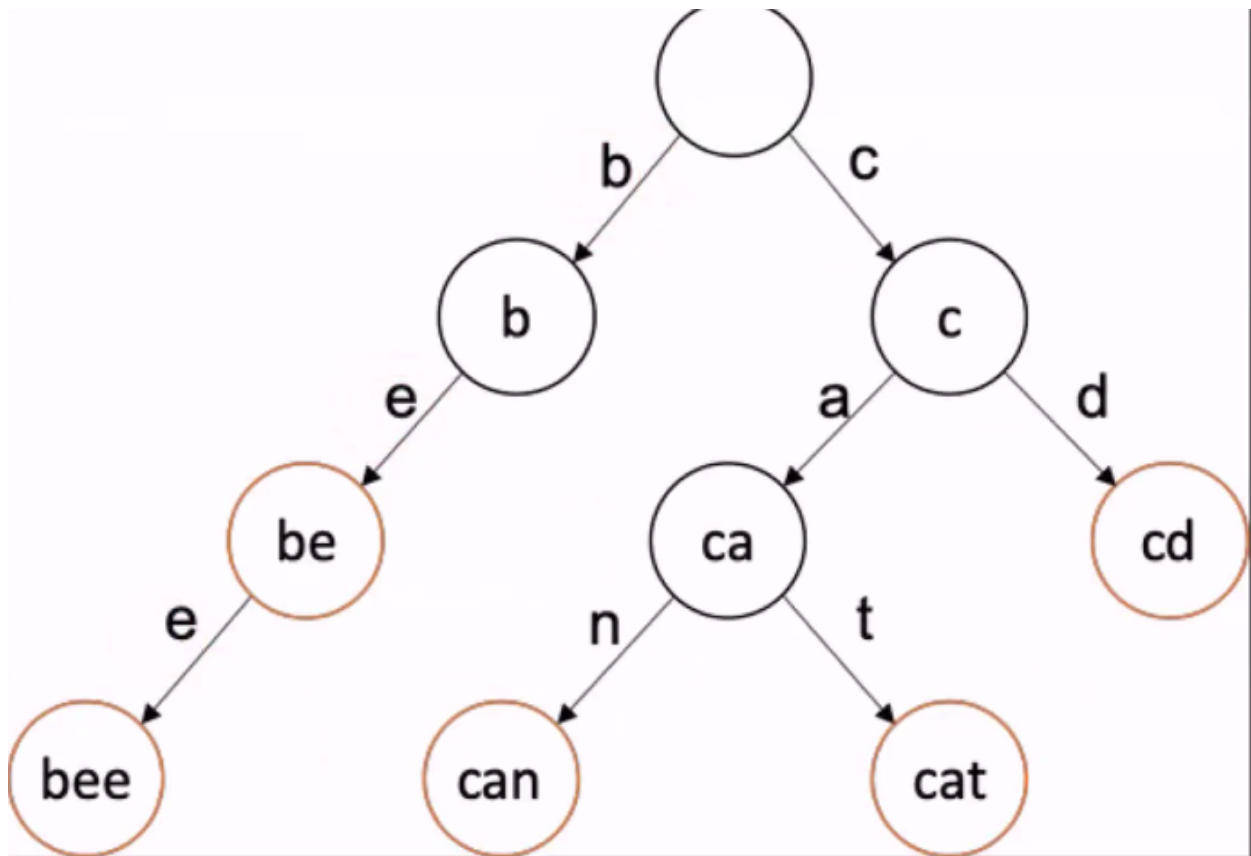


트라이(Trie)와 접두사 트리(Prefix Tree)의 차이점과 사용예시

practice 자료구조 · 2024. 10. 14. 15:13

트라이(Trie)와 접두사 트리(Prefix Tree)의 차이점과 사용예시

- 트라이와 접두사 트리는 동일한 자료구조를 나타냅니다.
- 이들은 문자열을 저장하고 검색하는 데 최적화된 트리로,
- 각 노드는 문자열의 한 문자와 연결된 자식 노드를 가집니다.
- 접두사 트리는 문자열의 접두사 공통성을 활용하여 검색 효율을 높입니다.



****트라이(Trie)****와 ****접두사 트리(Prefix Tree)****는 사실 동일한 개념을 가리킵니다. *****트라이*****는 *****접두사 트리*****의 다른 이름이며, 용어만 다를 뿐이지 그 구조와 기능은 동일합니다.

트라이(Trie)와 접두사 트리(Prefix Tree)의 관계

- ****Trie****는 ****Prefix Tree****의 또 다른 명칭입니다. "Trie"는 "Retrieval"에서 나온 것으로, 문자열 검색에 특화된 트리 구조라는 의미입니다. 트라이의 가장 큰 특징은 문자열의 공통 접두사를 ****효율적으로 저장하고 탐색****할 수 있다는 점입니다.
- ****Prefix Tree****라는 용어는 이 트리의 동작 방식에 주목한 표현으로, 각 경로가 문자열의 접두사를 형성하는 트리 구조를 의미합니다.

트라이(Trie) / 접두사 트리(Prefix Tree)의 주요 개념

- 각 노드는 문자열의 하나의 문자를 저장합니다.
- 루트에서 리프 노드로 가는 경로가 하나의 문자열을 나타냅니다.
- 문자열의 공통 접두사는 하나의 경로로 공유됩니다.
- 빠른 ****문자열 검색****, ****접두사 기반 검색****, ****자동 완성**** 등의 기능에 사용됩니다.

트라이의 주요 연산

1. ****삽입 (Insert)****: 문자열을 한 글자씩 트리의 각 노드에 삽입합니다. 이미 존재하는 접두사는 공유하고, 없는 부분만 새로 노드를 추가합니다.
2. ****검색 (Search)****: 입력된 문자열이 트라이에 존재하는지 확인하기 위해, 루트에서 시작하

여 문자열의 각 문자를 따라가며 탐색합니다.

3. ****접두사 검색 (Prefix Search)****: 특정 접두사로 시작하는 문자열들을 빠르게 검색할 수 있습니다.

트라이의 시간 복잡도

- ****삽입 및 검색****: 문자열의 길이가 (L) 일 때, 트라이에서 삽입 및 검색의 시간 복잡도는 $(O(L))$ 입니다.

- 이는 입력 문자열의 길이에 비례한 시간이 걸리며, 트리의 구조 덕분에 전체 문자열의 개수와는 관계없이 일정한 성능을 보장합니다.

트라이(Trie) / 접두사 트리(Prefix Tree)의 사용 예시

트라이는 문자열 검색과 관련된 다양한 애플리케이션에서 활용됩니다:

1. ****자동 완성 (Auto-complete)****: 사용자가 입력한 ****접두사****에 해당하는 모든 단어를 빠르게 제안할 수 있습니다. 예를 들어, 검색 엔진의 입력창에서 "app"을 입력하면 "apple", "application" 등 접두사가 "app"인 단어들을 제시할 수 있습니다.

2. ****사전(Dictionary) 구현****: 단어들이 저장된 트라이를 사용하여, 주어진 단어가 사전에 존재하는지 여부를 빠르게 확인할 수 있습니다.

3. ****문자열 집합에서 중복 검사****: 이미 저장된 문자열과 동일한 문자열이 있는지 빠르게 확인할 수 있습니다.

4. ****문자열 집합에서 패턴 검색****: 특정 패턴(특정 접두사를 포함한 문자열들)을 효율적으로 찾을 수 있습니다.



5. ****DNA 서열 분석****: 긴 DNA 서열을 분석할 때, 각 서열이 특정 패턴(접두사 또는 부분 문자열)을 포함하는지 빠르게 탐색하는 데 트라이가 사용될 수 있습니다.

6. ****IP 주소 라우팅****: 네트워크 라우터에서 접두사 기반으로 IP 주소를 관리하고 탐색할 때 트라이 구조를 사용하여 빠른 검색을 할 수 있습니다.

요약

- ****트라이(Trie)****와 ****접두사 트리(Prefix Tree)****는 동일한 자료구조를 가리키는 용어로, 문자열 검색과 접두사 기반 연산에 특화된 트리 구조입니다.

- 사용 사례로는 ****자동 완성****, ****사전 검색****, ****중복 문자열 탐색****, ****패턴 매칭**** 등이 있으며, 각 연산의 시간 복잡도는 문자열의 길이에 비례하여 효율적으로 동작합니다.

♡ 공감  

구독하기

'practice' 자료구조 카테고리의 다른 글

(Red-Black Tree)의 기본 속성과 그 왜곡 방지 방법 (0)

2024.10.14

B- / B+ 트리 차이점 (0)

2024.10.14

트라이(Trie) 자료구조의 기본 개념과 예시 (0)

2024.10.14

사이클 탐지 (0)

2024.10.14

그래프 탐색- DFS, BFS (0)

2024.10.14

관련글

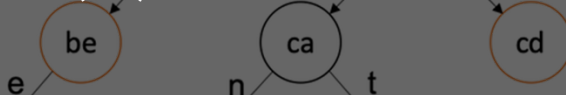
[관련글 더보기](#)

(Red-Black Tree)의 기본 속성과 그 왜곡
방지 방법

데이터 저장	모든 노드(내부 노드와 리프 노드)가 데이터를 저장함	데이터는 오직 리프 노드에만 저장됨
내부 노드	데이터와 포인터가 함께 저장됨	포인터만 저장되고, 데이터는 리프 노드에 저장됨
탐색 경로	데이터는 어디서든 찾을 수 있음	리프 노드에서만 찾을 수 있음
리프 노드 연결	리프 노드가 서로 연결되지 않음	리프 노드들이 Linked List로 연결되어 있음
순차 접근 성능	리프 노드 간에 직접 연결이 없으므로 순차 접근 성능이 낮음	리프 노드들이 연결되어 있어 순차 접근이 더 빠름

B- / B+ 트리 차이점

트라이(Trie) 자료구조의 기본 개념과 예시



사이클 탐지

자연어(NLP)

네이쳐2024 님의 블로그입니다.

구독하기 +



댓글 0



이름

비밀번호

내용을 입력하세요.



댓글
등록