

KMP 문자열 검색 알고리즘

practice 자료구조 · 2024. 10. 14. 15:18

KMP(Knuth-Morris-Pratt) 문자열 검색 알고리즘은 문자열 내에서 특정 패턴을 효율적으로 찾는 알고리즘입니다. 이 알고리즘은 **부분 일치**를 활용하여 불필요한 비교를 피하고, 패턴이 일치하지 않는 부분에서 이미 확인한 정보를 이용해 다음 비교를 효율적으로 수행합니다. 그 결과, KMP 알고리즘은 최악의 경우에도 시간 복잡도가 $O(n)$ 이 됩니다. n 은 본문 문자열의 길이입니다.

KMP 알고리즘의 주요 아이디어

KMP 알고리즘은 다음 두 단계로 나뉩니다:

- **부분 일치 테이블(PI 배열 또는 LPS 배열) 생성****: 패턴 내부의 접두사와 접미사의 정보를 저장하여, 일치하지 않는 경우 얼마나 건너뛰어야 하는지를 계산하는 데 사용됩니다.
- **본문 문자열 탐색****: 본문 문자열에서 패턴을 비교하며, 일치하지 않는 경우 부분 일치 테이블을 사용해 다시 처음부터 비교하지 않고 효율적으로 진행합니다.

KMP 알고리즘의 동작 방식

1. **부분 일치 테이블(PI 배열 또는 LPS 배열) 생성**

- 이 테이블은 패턴의 각 위치까지의 ****접두사****와 ****접미사****가 일치하는 최대 길이를 기록합니다.
- 예를 들어, 패턴의 앞부분에서 특정 부분 문자열이 다시 나타나면, 그 접두사 정보를 사용해 불필요한 반복 비교를 피할 수 있습니다.

****예시****: 패턴이 `"ABABCABAB"`일 때, 각 접두사와 접미사가 일치하는 부분을 기록한 테이블은 다음과 같습니다:

문자 위치	패턴	LPS 값 (부분 일치 길이)
0	A	0
1	B	0
2	A	1
3	B	2
4	C	0
5	A	1
6	B	2
7	A	3
8	B	4

위 표에서, (LPS[i])는 패턴의 0부터 i까지의 부분 문자열에서 **접두사**와 **접미사**가 일치하는 최대 길이를 나타냅니다.

2. 본문 문자열 탐색

- 본문 문자열에서 패턴을 탐색할 때, 패턴의 일부분이 일치하다가 불일치가 발생하면, 패턴을 **부분 일치 테이블**을 참조하여 건너뛰어 다시 비교합니다. 이렇게 함으로써 불필요한 비교를 피할 수 있습니다.

예시:

- 본문: "ABABDABACDABABCABAB"

- 패턴: "ABABCABAB"

1. 처음부터 비교를 시작해 패턴이 본문과 일치할 때까지 비교합니다.
2. 일치하다가 중간에 불일치가 발생하면, 부분 일치 테이블을 이용해 패턴의 다음 위치로 점프하여 다시 비교합니다.

이 과정을 통해 중복된 부분에 대해 다시 처음부터 비교하지 않으므로, KMP 알고리즘의 시간 복잡도는 $O(n)$ 으로 유지됩니다.

KMP 알고리즘의 시간 복잡도

- **부분 일치 테이블 생성**: $O(m)O(m)O(m)$, 여기서 mmm은 패턴의 길이

- **본문 탐색:** $O(n)O(n)O(n)$, 여기서 nnn 은 본문 문자열의 길이

따라서, 전체 시간 복잡도는 $O(n+m)O(n + m)O(n+m)$ 입니다.

KMP 알고리즘의 장점



1. ****효율적인 탐색**:** 이미 비교한 부분을 다시 비교하지 않기 때문에 불필요한 연산을 줄입니다.
2. ****최악의 경우에도 $O(n)$:** 본문 문자열의 길이에 비례하는 시간이 걸리며, 매우 효율적입니다.
3. ****접두사 정보를 활용**:** 패턴 내의 중복된 부분 정보를 사용해 탐색을 효율화합니다.

KMP 알고리즘의 사용 예

- ****텍스트 편집기**:** 특정 단어 또는 구문을 텍스트 내에서 빠르게 찾기 위한 기능
- ****문자열 검색 엔진**:** 대규모 데이터에서 특정 패턴을 검색하는 경우
- ****바이오인포매틱스**:** DNA 서열 분석에서 특정 패턴을 찾아내는 작업

요약

- ****KMP 알고리즘****은 본문에서 패턴을 효율적으로 찾는 알고리즘으로, ****부분 일치 테이블 (LPS 배열)****을 사용하여 불필요한 중복 비교를 피합니다.
- ****시간 복잡도****는 $O(n+m)$ 으로 매우 효율적이며, 음수 가중치나 무한 루프 없이 안정적으로 작동합니다.
- 문자열 검색, 패턴 매칭 등의 문제에서 널리 사용됩니다.

♡ 공감  

구독하기

'practice' 자료구조 카테고리의 다른 글

트라이(Trie)와 접두사 트리(Prefix Tree)의 차이점과 사용예시 (0)	2024.10.14
(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

관련글

[관련글 더보기](#)

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

(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	데이터는 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

자연어(NLP)

네이쳐2024 님의 블로그입니다.

구독하기 +

댓글 0

이름

비밀번호

내용을 입력하세요.

등록