트라이 (Trie)

01 트라이 이론

02 트라이의 활용

신 제 용

01 트라이 이론

단어를 검색하는 데에 최적화된 트라이 자료구조와 이를 구현하는 방법을 학습합니다.

학습 키워드 - 트라이, 검색, 자료구조

Chapter 01
트라이 이론

ZEROBASE SCHOOL

트라이 (Trie)

- 검색 트리(Re**trie**val tree)

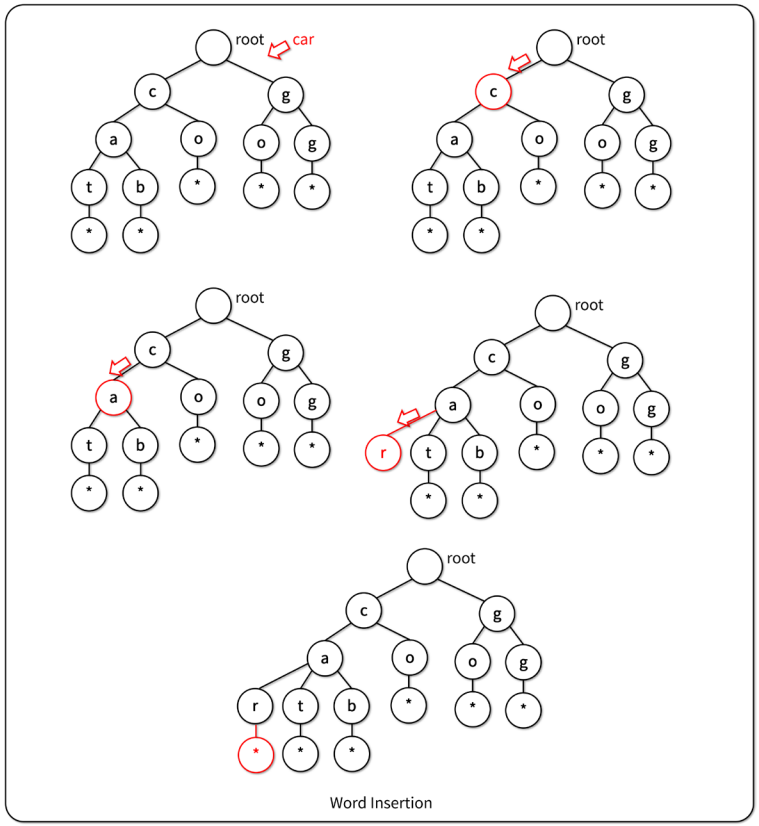
- 문자열을 저장하고 빠르게 탐색하기 위한 자료 구조

- 트라이를 구성하기 위한 별도의 메모리가 필요하나, 매우 빠른 탐색이 가능

  → 단순 문자열 탐색: $O(MN)$, 트라이에서 탐색: $O(M)$

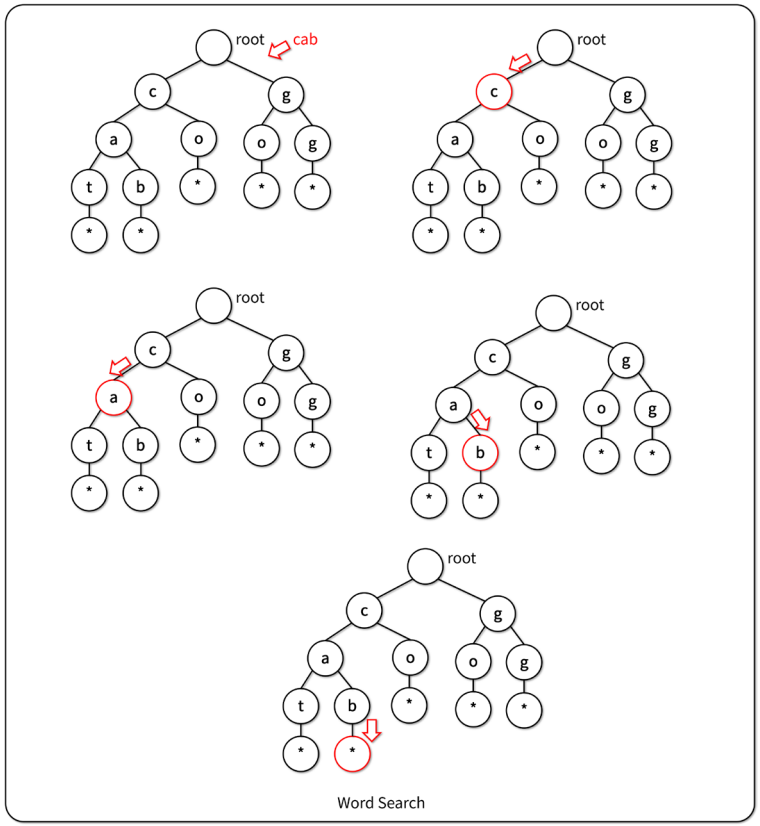
  (M: 문자열의 길이, N: 문자열의 개수)

자료의 입력



- 각 노드는 문자열의 각 문자를 의미
- 루트 노드에서 시작하여, 문자열의 문자를 순서대로 자식 노드로 생성
- 기존에 같은 문자를 가진 자식 노드가 있다면, 새 노드를 생성하지 않고 해당 노드를 사용
- 모든 문자열 입력이 끝나면, 마지막에 END문자(*)를 입력

자료의 탐색



- 루트 노드부터 시작하여, 문자열의 각 문자를 하나씩 자식 노드에서 찾아간다.
- 자식 노드 중에 문자가 없으면 해당 문자열은 트라이에 존재하지 않는다.
- 모든 문자를 탐색한 후에, END문자(*)까지 찾으면 해당 문자열은 트라이에 존재한다.

02 트라이의 활용

다음 챕터에서는 트라이를 활용하는 예시를 확인하고 구현해봅니다.

Chapter 01
트라이 이론

02 트라이의 활용

트라이를 활용하는 기초 유형 문제를 확인하고 직접 풀어봅시다.

학습 키워드 - 트라이, 활용, 구현

Chapter 02
트라이의 활용

ZEROBASE SCHOOL

```python
# Trie를 자료구조를 구현하시오.

class Node:
    def __init__(self):
        self.children = dict()


class Trie:
    def __init__(self):
        self.root = Node()

    def add_word(self, word):
        pass

    def search(self, query):
        pass
```

Chapter 02
트라이의 활용

ZEROBASE SCHOOL

```python
# Prob1.
#
# 와일드카드 검색기
# 검색할 단어가 리스트 words에 주어져 있다.
# 이 words에 queries에 있는 단어로 시작하는 단어를 모두 찾으려 한다.
# 예를 들어, words = ["buy", "bull", "bucket", "bill"] 일 때,
# "bu"를 검색하면 "bu"로 시작하는 단어 "buy", "bull", "bucket"이 일치하여 답은
3이 된다.
# 이 프로그램을 구현하시오.
#
# 예시 입출력
# words = ["fast", "zero", "base", "study", "baseball", "steel"]
# queries = ["fa", "ba", "zer", "st", "sti"]
# 출력: [1, 2, 1, 2, 0]


def solution(words, quries):
    pass


if __name__ == '__main__':
    words = ["fast", "zero", "base", "study", "baseball", "steel"]
    queries = ["fa", "ba", "zer", "st", "sti"]
    sol = solution(words, queries)
    print(sol)
```

```python
# Prob2.
#
# 와일드카드 검색기2
# 검색할 단어가 리스트 words에 주어져 있다.
# 이번에는 queries의 검색 단어가 와일드카드 ?를 포함하고 있다.
# 모든 검색 단어는 마지막에 최소 한개의 ?를 가지고 있으며,
# ?의 개수만큼 아무 문자나 매칭이 가능하다.
# 이 프로그램을 구현하시오.
#
# 예시 입출력
# words = ["fast", "zero", "base", "study", "baseball", "steel"]
# queries = ["fa??", "ba??", "ze?", "st???", "z???"]
# 출력: [1, 1, 0, 2, 1]

def solution(words, queries):
    pass

if __name__ == '__main__':
    words = ["fast", "zero", "base", "study", "baseball", "steel"]
    queries = ["fa??", "ba??", "ze?", "st???", "z???"]
    sol = solution(words, queries)
    print(sol)
```