Trie

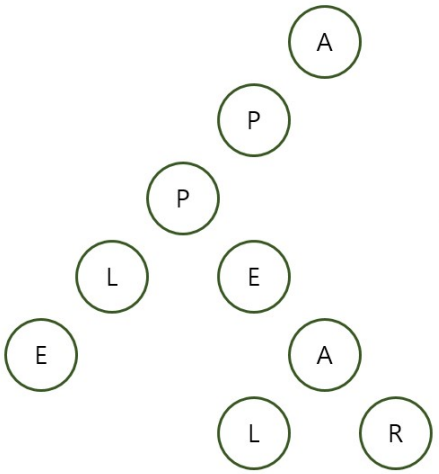
Trie는 어떻게 보면 그래프라고 할 수 있지만 자연어 처리에서 문자열 탐색을 위해 많이 쓰이는 알고리즘이라고 하여 문자열 탐색 알고리즘에 넣었다. 기본적인 원리는 딕셔너리 안의 딕셔너리 안의 딕셔너리로 구성되어 어떠한 문자열이 주어졌을 때 그 문자열을 구성하는 모든 문자열이 딕셔너리의 끝까지 갔을때 True 상태를 유지하면 그 문자열이 Trie 자료 구조 안에 있는 것으로 판단하는 것이다.

코드

```
class TrieNode:

    def __init__(self):
        self.word = False
        self.children = {}

class Trie:

    def __init__(self):
        self.root = TrieNode()

    def insert(self, word: str) -> None:

        node = self.root

        for char in word:
            if char not in node.children:
                node.children[char] = TrieNode()
            node = node.children[char]
        node.word = True

    def search(self, word: str) -> bool:
        node = self.root

        for char in word:
            if char not in node.children:
                return False
            node = node.children[char]

        return node.word

    def startsWith(self, prefix: str) -> bool:
```

```
        node = self.root

        for char in prefix:
            if char not in node.children:
                return False
            node = node.children[char]

        return True

trie = Trie()

trie.insert('apple')
trie.insert('appear')
print(trie.search('apple'))

print(trie.root.children)
```

Trie의 형태



"A": { "P": { "P": {"L": {"E": True}, "E": {"A": {"L": True, "R": True} } } }