```python
class CompressedTrieNode:
    def __init__(self):
        self.children = {}
        self.is_end_of_word = False

class CompressedTrie:
    def __init__(self):
        self.root = CompressedTrieNode()

    def insert(self, word):
        node = self.root
        i = 0
        while i < len(word):
            for label in node.children.keys():
                 common_length = self.common_prefix_length(label, word[i:])
                if common_length > 0:
                    if common_length == len(label):
                        node = node.children[label]
                        i += common_length
                    else:
                         remaining_label = label[common_length:]
                        new_node = CompressedTrieNode()
                         new_node.children[remaining_label] = node.children.pop(label)
                         node.children[label[:common_length]] = new_node
                        node = new_node
                        i += common_length
                    break
            else:
                node.children[word[i:]] = CompressedTrieNode()
                node = node.children[word[i:]]
                i = len(word)
        node.is_end_of_word = True

    def search(self, word):
        node = self.root
        i = 0
        while i < len(word):
            found = False
            for label in node.children.keys():
                 common_length = self.common_prefix_length(label, word[i:])
                if common_length == len(label):
                    node = node.children[label]
                    i += common_length
                    found = True
                    break
                elif common_length > 0:
                    return False
            if not found:
                return False
        return node.is_end_of_word

    def common_prefix_length(self, s1, s2):
        min_len = min(len(s1), len(s2))
        for i in range(min_len):
            if s1[i] != s2[i]:
                  t i
                return i
        return min_len

if __name__ == "__main__":
    compressed_trie = CompressedTrie()
    words = ["apple", "app", "bat", "ball", "batman"]

    for word in words:
        compressed_trie.insert(word)
```

```
print(compressed_trie.search("apple"))
print(compressed_trie.search("bat"))
print(compressed_trie.search("batman"))
print(compressed_trie.search("ape"))
```

```
 True
 True
 True
 False
```