

Competitive programming

Assignment-05

2303A51463

Batch-07

1: Building and Utilizing Tries for String Problems

Problem: Library Book Title Search Using Trie

A digital library stores a list of book titles.

When a student types the first few letters of a title, the system immediately displays matching books.

Since many book titles begin with common prefixes, the library uses a Trie (Prefix Tree) to efficiently:

- Store book titles
- Search for a complete title
- Check whether any title starts with a given prefix

Task

Write a program using a Trie to support:

- Insert operation: Add a book title
- Search operation: Check whether a book title exists
- Prefix query: Check whether any book title starts with a given prefix

Example Test Case 3

Input

- Number of book titles: 5
- Book titles:
• ["Data", "Database", "DataScience", "Design", "Development"]

Operations:

1. Search for book "Data"

2. Search for book "DataMining"

3. Check prefix "Data"

4. Check prefix "Dev"

Output

- Book "Data" found → True
- Book "DataMining" not found → False
- Prefix "Data" exists → True
- Prefix "Dev" exists → True

Explanation

Book titles inserted into the Trie

Data, Database, DataScience, Design, Development

Conceptual Trie Structure

26th Feb,

2025

5:00PM

(root)

/ \

D (other)

|

a

|

t

|

a*

/ \

b S

||

a c

||

s i

||

e e

|

(end)

From 'D'

|

e

|

s

|

i

|

g

|

n*

|

v

|

e

|

|

|

o

|

p

|

m

|

e

|

n

|

t*

(* indicates end of a complete word)

Operation 1: Search "Data"

Characters D → a → t → a exist

End-of-word marker is present

Book found

Operation 2: Search "DataMining"

Characters D → a → t → a exist

Character M does not continue the path

Book not found

Operation 3: Prefix "Data"

Characters D → a → t → a exist

Matching titles:

- Data
- Database
- DataScience

Prefix exists

Operation 4: Prefix "Dev"

Characters D → e → v exist

Matching title:

- Development

Prefix exists

Algorithm:

1. Set current = root.
2. For each character c in title:

- Convert c to lowercase.
- Compute index = c - 'a'.
- If current.children[index] is NULL:
 - Create a new Trie node there.
- Move current to current.children[index].

3. After the last character:

- Mark current.isEndOfWord = true.

Code:

```
import java.util.Scanner;

class TrieNode {
    TrieNode[] children;
    boolean isEndOfWord;
    TrieNode() {
        children = new TrieNode[26];
        isEndOfWord = false;
    }
}

class Trie {
    private TrieNode root;
    Trie() {
        root = new TrieNode();
    }
    public void insert(String word) {
        TrieNode current = root;
        for (int i = 0; i < word.length(); i++) {
            char ch = Character.toLowerCase(word.charAt(i));
            if (ch < 'a' || ch > 'z') continue;
            int index = ch - 'a';
            if (current.children[index] == null) {
```

```

        current.children[index] = new TrieNode();
    }
    current = current.children[index];
}
current.isEndOfWord = true;
}

public boolean search(String word) {
    TrieNode current = root;
    for (int i = 0; i < word.length(); i++) {
        char ch = Character.toLowerCase(word.charAt(i));
        if (ch < 'a' || ch > 'z') continue;
        int index = ch - 'a';
        if (current.children[index] == null) {
            return false;
        }
        current = current.children[index];
    }
    return current.isEndOfWord;
}

public boolean startsWith(String prefix) {
    TrieNode current = root;
    for (int i = 0; i < prefix.length(); i++) {
        char ch = Character.toLowerCase(prefix.charAt(i));
        if (ch < 'a' || ch > 'z') continue;
        int index = ch - 'a';
        if (current.children[index] == null) {
            return false;
        }
        current = current.children[index];
    }

```

```

    }
    return true;
}
}

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        Trie trie = new Trie();

        int n = sc.nextInt();

        sc.nextLine();

        for (int i = 0; i < n; i++) {

            String title = sc.nextLine();

            trie.insert(title);

        }

        int q = sc.nextInt();

        sc.nextLine();

        for (int i = 0; i < q; i++) {

            String line = sc.nextLine();

            String[] parts = line.split(" ");

            String operation = parts[0];

            String value = parts[1];

            if (operation.equalsIgnoreCase("search")) {

                System.out.println(trie.search(value));

            }

            else if (operation.equalsIgnoreCase("prefix")) {

                System.out.println(trie.startsWith(value));

            }

        }

    }

}

```

```

    }

    sc.close();
}

}

```

```

1  import java.util.Scanner;
2  class TrieNode {
3      TrieNode[] children;
4      boolean isEndOfWord;
5      TrieNode() {
6          children = new TrieNode[26];
7          isEndOfWord = false;
8      }
9  }
10 class Trie {
11     private TrieNode root;
12     Trie() {
13         root = new TrieNode();
14     }
15     public void insert(String word) {
16         TrieNode current = root;
17         for (int i = 0; i < word.length(); i++) {
18             char ch = Character.toLowerCase(word.charAt(i));
19             if (ch < 'a' || ch > 'z') continue;
20             int index = ch - 'a';
21             if (current.children[index] == null) {
22                 current.children[index] = new TrieNode();
23             }
24             current = current.children[index];
25         }
26         current.isEndOfWord = true;
27     }
28     public boolean search(String word) {
29         TrieNode current = root;
30         for (int i = 0; i < word.length(); i++) {
31             char ch = Character.toLowerCase(word.charAt(i));
32             if (ch < 'a' || ch > 'z') continue;
33             int index = ch - 'a';
34             if (current.children[index] == null) {
35                 return false;
36             }
37             current = current.children[index];
38         }
39         return current.isEndOfWord;
40     }
41 }

```

STDIN

```

5
Data
Database
DataScience
Design
Development
4
search Data
search DataMining
prefix Data
prefix Dev

```

Output: 41 ms | 46.0 MB

```

true
false
true
true

```

```

40     return current.isEndOfWord;
41 }
42 public boolean startsWith(String prefix) {
43     TrieNode current = root;
44     for (int i = 0; i < prefix.length(); i++) {
45         char ch = Character.toLowerCase(prefix.charAt(i));
46         if (ch < 'a' || ch > 'z') continue;
47         int index = ch - 'a';
48         if (current.children[index] == null) {
49             return false;
50         }
51         current = current.children[index];
52     }
53     return true;
54 }
55 public class Main {
56     public static void main(String[] args) {
57         Scanner sc = new Scanner(System.in);
58         Trie trie = new Trie();
59         int n = sc.nextInt();
60         sc.nextLine();
61         for (int i = 0; i < n; i++) {
62             String title = sc.nextLine();
63             trie.insert(title);
64         }
65         int q = sc.nextInt();
66         sc.nextLine();
67         for (int i = 0; i < q; i++) {
68             String line = sc.nextLine();
69             String[] parts = line.split(" ");
70             String operation = parts[0];
71             String value = parts[1];
72             if (operation.equalsIgnoreCase("search")) {
73                 System.out.println(trie.search(value));
74             }
75             else if (operation.equalsIgnoreCase("prefix")) {

```

STDIN

```

5
Data
Database
DataScience
Design
Development
4
search Data
search DataMining
prefix Data
prefix Dev

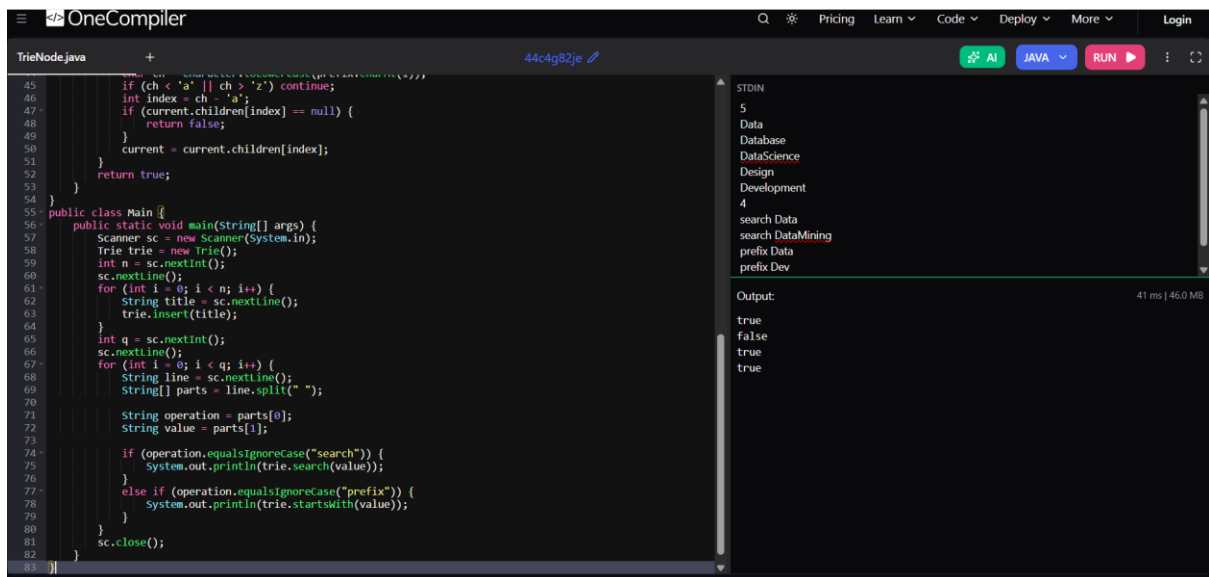
```

Output: 41 ms | 46.0 MB

```

true
false
true
true

```



```
45         if (ch < 'a' || ch > 'z') continue;
46         int index = ch - 'a';
47         if (current.children[index] == null) {
48             return false;
49         }
50         current = current.children[index];
51     }
52     return true;
53 }
54 }
55 public class Main {
56     public static void main(String[] args) {
57         Scanner sc = new Scanner(System.in);
58         Trie trie = new Trie();
59         int n = sc.nextInt();
60         sc.nextLine();
61         for (int i = 0; i < n; i++) {
62             String title = sc.nextLine();
63             trie.insert(title);
64         }
65         int q = sc.nextInt();
66         sc.nextLine();
67         for (int i = 0; i < q; i++) {
68             String line = sc.nextLine();
69             String[] parts = line.split(" ");
70
71             String operation = parts[0];
72             String value = parts[1];
73
74             if (operation.equalsIgnoreCase("search")) {
75                 System.out.println(trie.search(value));
76             }
77             else if (operation.equalsIgnoreCase("prefix")) {
78                 System.out.println(trie.startWith(value));
79             }
80         }
81         sc.close();
82     }
83 }
```

STDIN

5
Data
Database
DataScience
Design
Development
4
search Data
search DataMining
prefix Data
prefix Dev

Output: 41 ms | 46.0 MB

true
false
true
true

2.Problem: Mobile Contact Name Search Using Trie

Problem Statement

A mobile phone stores a list of contact names.

When a user types the first few letters of a contact name, the phone should instantly check whether matching contacts exist.

Since many contact names share common prefixes, the mobile phone uses a Trie (Prefix Tree) to efficiently:

- Store contact names
- Search for a complete contact name
- Check whether any contact name starts with a given prefix

Task

Write a program using a Trie to support the following operations:

- Insert Operation

Add a contact name to the Trie.

- Search Operation

Check whether a complete contact name exists in the Trie.

- Prefix Query Operation

Check whether any contact name starts with a given prefix.

Example Test Case

Input

- Number of contact names: 5
- Contact names:
["Anil", "Anita", "Anand", "Suresh", "Sunil"]

Operations:

1. Search for contact "Anil"
2. Search for contact "Anitha"
3. Check prefix "Ani"
4. Check prefix "Su"

Output

- Contact "Anil" found → True
- Contact "Anitha" not found → False
- Prefix "Ani" exists → True
- Prefix "Su" exists → True

Explanation

Contact Names Inserted into the Trie

- Anil
- Anita
- Anand
- Suresh
- Sunil

Conceptual Trie Structure

(root)

/ \

A S

| |

n u

/ \ \

i a r

/\|\

l* t* n e

||

d* s

|

h*

(from $S \rightarrow u \rightarrow n \rightarrow i \rightarrow l^*$)

* indicates the end of a complete contact name.

Operation-wise Explanation

Operation 1: Search "Anil"

- Characters followed: $A \rightarrow n \rightarrow i \rightarrow l$
- End-of-word marker found

Contact found

Operation 2: Search "Anitha"

- Characters $A \rightarrow n \rightarrow i \rightarrow t$ exist
- Character h does not continue the Trie path

Contact not found

Operation 3: Prefix Query "Ani"

- Characters $A \rightarrow n \rightarrow i$ exist
- Matching contacts:

o Anil

o Anita

Prefix exists

Operation 4: Prefix Query "Su"

- Characters $S \rightarrow u$ exist
- Matching contacts:

o Suresh

o Sunil

Prefix exists

Algorithm 1: INSERT(ContactName)

Input: String name

Steps:

1. Set current = root.
2. For each character c in name:
 - Convert to lowercase.
 - $\text{index} = c - 'a'$
 - If current.children[index] is NULL:
 - Create a new node.
 - Move current to that child.
3. After the last character:
 - Mark current.isEndOfWord = true.

Algorithm 2: SEARCH(ContactName)

Input: String name

Output: True if contact exists, else False

Steps:

1. Start at root.
2. For each character:
 - Convert to lowercase.
 - Compute index.
 - If child does not exist → return False.
3. After traversal:
 - If isEndOfWord is true → return True
 - Else → return False.

Algorithm 3: PREFIX QUERY

Input: String prefix

Output: True if any contact starts with prefix, else False

Steps:

1. Start at root.
2. Traverse characters of prefix.
3. If any character path is missing → return False.
4. If all matched → return True.

Code:

```
import java.util.Scanner;

class TrieNode {
    TrieNode[] children = new TrieNode[26];
    boolean isEndOfWord;
}

class Trie {
    private TrieNode root;

    Trie() {
        root = new TrieNode();
    }

    public void insert(String word) {
        TrieNode current = root;
        for (char ch : word.toLowerCase().toCharArray()) {
            if (ch < 'a' || ch > 'z') continue;
            int index = ch - 'a';
            if (current.children[index] == null) {
                current.children[index] = new TrieNode();
            }
            current = current.children[index];
        }
    }
}
```

```

        current.isEndOfWord = true;
    }

    public boolean search(String word) {
        TrieNode current = root;
        for (char ch : word.toLowerCase().toCharArray()) {
            if (ch < 'a' || ch > 'z') continue;
            int index = ch - 'a';
            if (current.children[index] == null) {
                return false;
            }
            current = current.children[index];
        }
        return current.isEndOfWord;
    }

    public boolean startsWith(String prefix) {
        TrieNode current = root;
        for (char ch : prefix.toLowerCase().toCharArray()) {
            if (ch < 'a' || ch > 'z') continue;
            int index = ch - 'a';
            if (current.children[index] == null) {
                return false;
            }
            current = current.children[index];
        }
        return true;
    }
}

public class Main {
    public static void main(String[] args) {

```

```
Scanner sc = new Scanner(System.in);
Trie trie = new Trie();
int n = sc.nextInt();
sc.nextLine();
for (int i = 0; i < n; i++) {
    String name = sc.nextLine();
    trie.insert(name);
}
int q = sc.nextInt();
sc.nextLine();
for (int i = 0; i < q; i++) {
    String[] parts = sc.nextLine().split(" ");
    String operation = parts[0];
    String value = parts[1];
    if (operation.equalsIgnoreCase("search")) {
        System.out.println(trie.search(value));
    }
    else if (operation.equalsIgnoreCase("prefix")) {
        System.out.println(trie.startsWith(value));
    }
}
sc.close();
}
```

OneCompiler

Q * Pricing Learn Code Deploy More Login

TrieNode.java

44c4g82je

AI JAVA RUN

```
1 import java.util.Scanner;
2 class TrieNode {
3     TrieNode[] children = new TrieNode[26];
4     boolean isEndOfWord;
5 }
6 class Trie {
7     private TrieNode root;
8     Trie() {
9         root = new TrieNode();
10    }
11    public void insert(String word) {
12        TrieNode current = root;
13        for (char ch : word.toLowerCase().toCharArray()) {
14            if (ch < 'a' || ch > 'z') continue;
15            int index = ch - 'a';
16            if (current.children[index] == null) {
17                current.children[index] = new TrieNode();
18            }
19            current = current.children[index];
20        }
21        current.isEndOfWord = true;
22    }
23    public boolean search(String word) {
24        TrieNode current = root;
25        for (char ch : word.toLowerCase().toCharArray()) {
26            if (ch < 'a' || ch > 'z') continue;
27            int index = ch - 'a';
28            if (current.children[index] == null) {
29                return false;
30            }
31            current = current.children[index];
32        }
33        return current.isEndOfWord;
34    }
35    public boolean startsWith(String prefix) {
36        TrieNode current = root;
37        for (char ch : prefix.toLowerCase().toCharArray()) {
38            if (ch < 'a' || ch > 'z') continue;
39            int index = ch - 'a';
```

STDIN

Anil
Anita
Anand
Suresh
Sunil
4
search Anil
search Anitha
prefix Ani
prefix Su

Output: 49 ms | 45.7 MB

true
false
true
true

OneCompiler

Q * Pricing Learn Code Deploy More Login

TrieNode.java

44c4g82je

AI JAVA RUN

```
36     TrieNode current = root;
37     for (char ch : prefix.toLowerCase().toCharArray()) {
38         if (ch < 'a' || ch > 'z') continue;
39         int index = ch - 'a';
40         if (current.children[index] == null) {
41             return false;
42         }
43         current = current.children[index];
44     }
45     return true;
46 }
47 }
48 public class Main {
49     public static void main(String[] args) {
50         Scanner sc = new Scanner(System.in);
51         Trie trie = new Trie();
52         int n = sc.nextInt();
53         sc.nextLine();
54         for (int i = 0; i < n; i++) {
55             String name = sc.nextLine();
56             trie.insert(name);
57         }
58         int q = sc.nextInt();
59         sc.nextLine();
60         for (int i = 0; i < q; i++) {
61             String[] parts = sc.nextLine().split(" ");
62             String operation = parts[0];
63             String value = parts[1];
64             if (operation.equalsIgnoreCase("search")) {
65                 System.out.println(trie.search(value));
66             }
67             else if (operation.equalsIgnoreCase("prefix")) {
68                 System.out.println(trie.startsWith(value));
69             }
70         }
71         sc.close();
72     }
73 }
74 }
```

STDIN

Anil
Anita
Anand
Suresh
Sunil
4
search Anil
search Anitha
prefix Ani
prefix Su

Output: 49 ms | 45.7 MB

true
false
true
true