PDP ONLINE

DSA – Data Structures

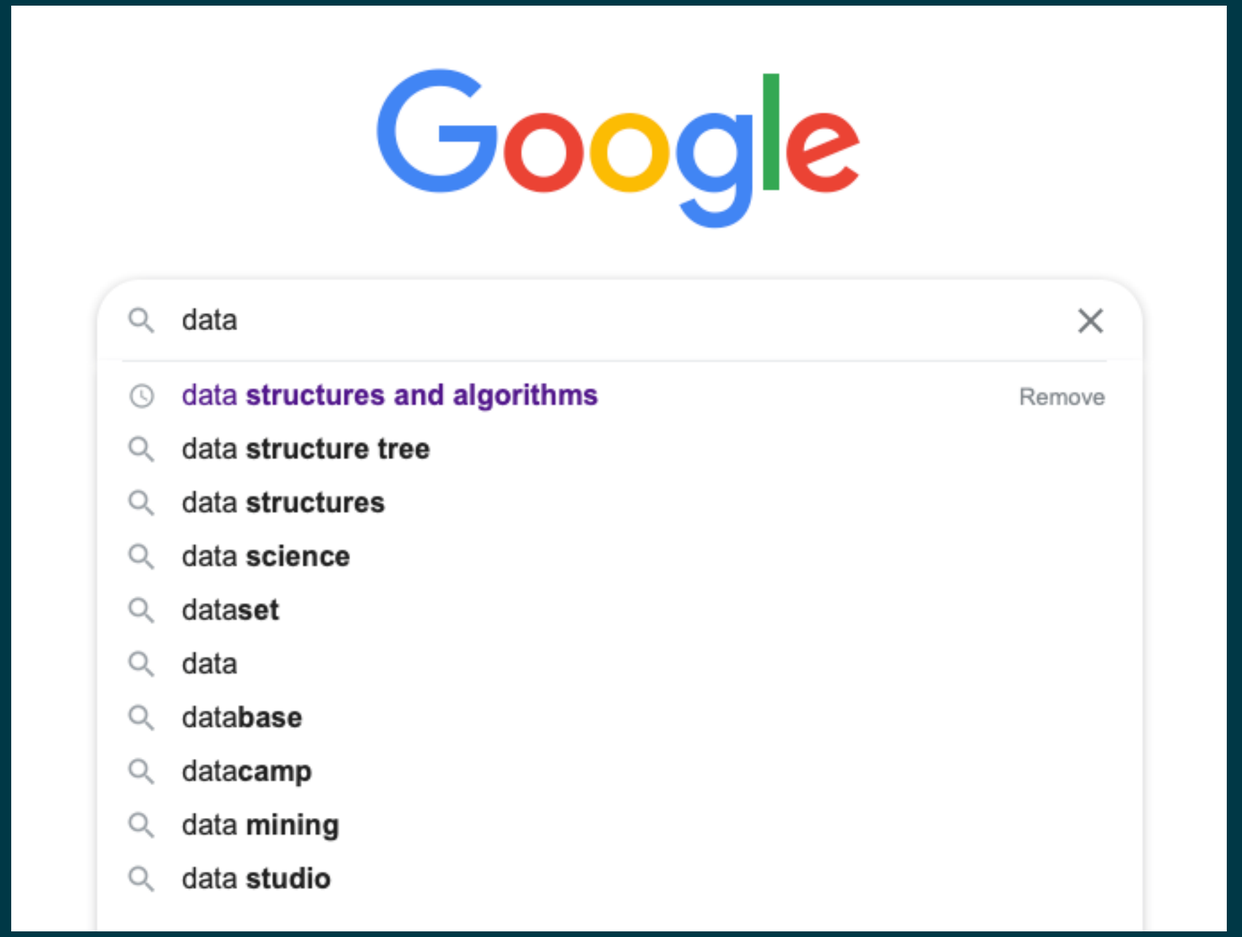
Trie

Course Planning

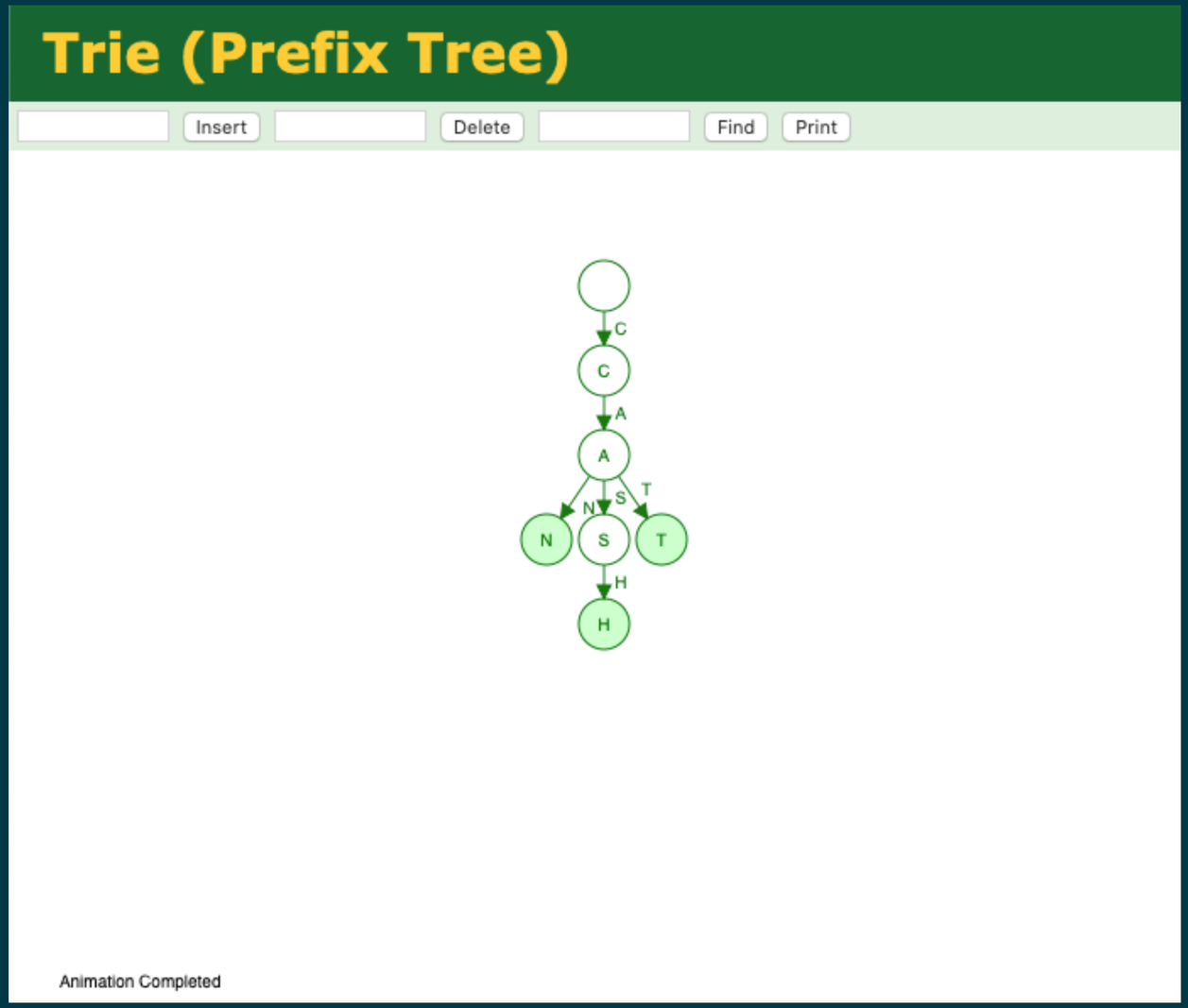
Algorithms	Data Structures	Algorithmic Approaches	Interview Practices
1.Introduction	1.Asymptotic Analysis	1.Search Algorithms	1.In-place Reversal
2.Number 1	2.Dynamic Array	2.Sort Algorithms	2.Two Heaps
3.Number 2	3.LinkedList	3.Dac Algorithms	3.Subsets
4.String 1	4.Stack	4.Recursion	4.Modified BS
5.String 2	5.Queue	5.Sliding Window	5.Bitwise XOR
6.Array 1	6.HashTable	6.Two Pointers	6.Top 'K' Elements
7.Array 2	7.Tree	7.Fast & Slow	7.K-way Merge
8.Matrix	8.Trie	8.Cyclic Sort	8.Knapsack Problem
9.DP 1	9.Directed Graph	9.Breadth First Search	9.Topological Sort
10.DP 2	10.Undirected Graph	10.Depth First Search	10.Mock Interview

https://kurbanovxurshid.medium.com/data-structures-and-algorithms-course-plan-1ab912ec1e17

Implementation

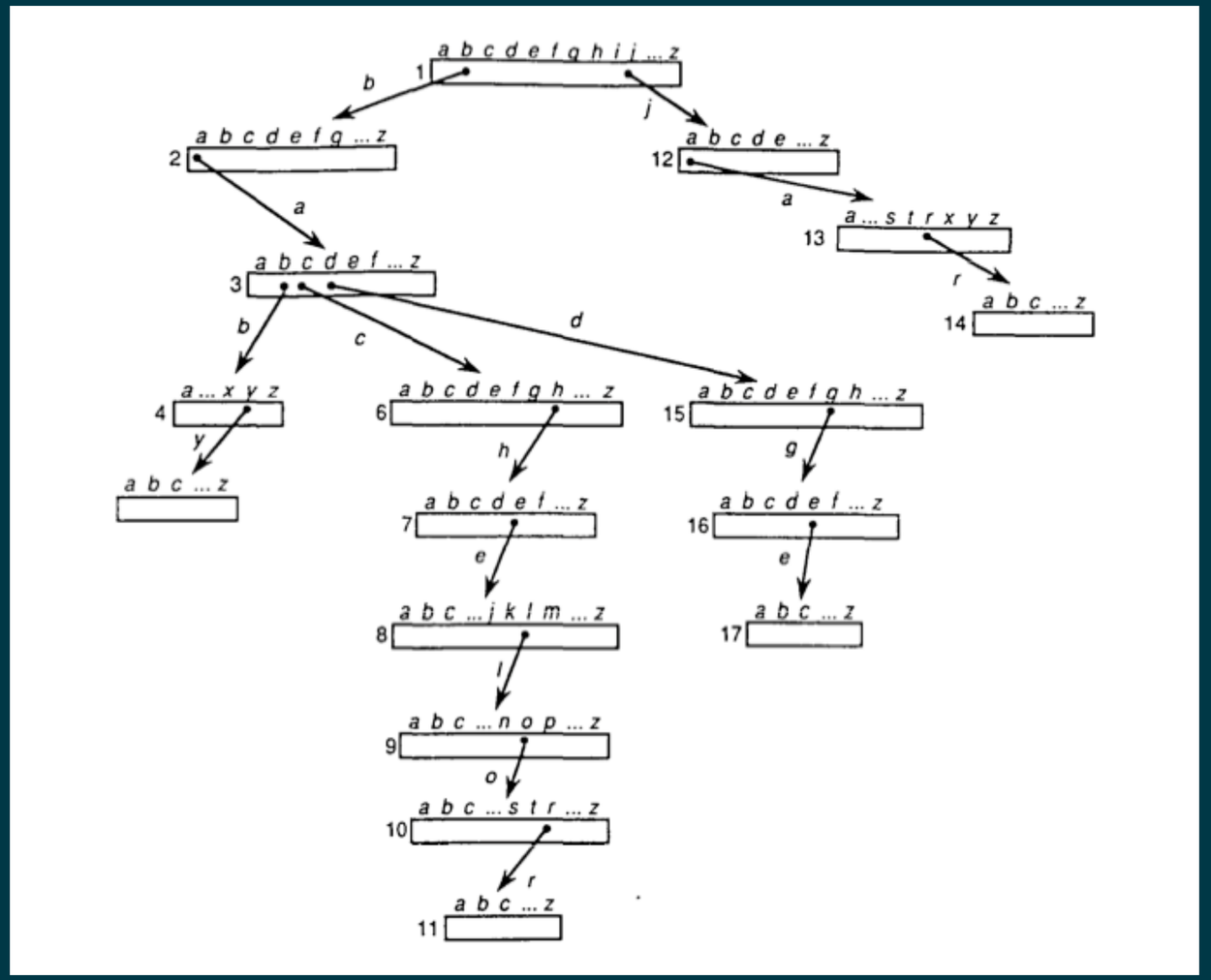


Visualization



https://www.cs.usfca.edu/~galles/visualization/Trie.html

Structure


a b c d e f g h i j ... z
1
b
j
a b c d e f g ... z
2
a b c d e ... z
12
a
a
a ... s t r x y z
13
r
a b c ... z
14
a
a b c d e f ... z
3
b
c
d
a ... x y z
4
a b c d e f g h ... z
6
a b c d e f g h ... z
15
y
h
g
a b c ... z
a b c d e f ... z
7
a b c d e f ... z
16
e
e
a b c ... j k l m ... z
8
a b c ... z
17
l
a b c ... n o p ... z
9
o
a b c ... s t r ... z
10
r
a b c ... z
11

Trie with Array

```java
public class TrieArray {

    public class Node{
        private char value;
        private boolean isWord;
        private Node[] children;

        public Node(char value){
            this.value = value;
            this.children = new Node[26];
            this.isWord = false;
        }
    }

    Node root;

    public TrieArray() {
        root  = new Node('\0');
    }
```

insert

```java
public void insert(String word) {
    Node curr = root;
    for(int i =0;i<word.length();i++){
        char c = word.charAt(i);
        int index = c-'a';

        if(curr.children[index] == null)
            curr.children[index] = new Node(c);

        curr = curr.children[index];
    }
    curr.isWord = true;
}
```

search

```java
public boolean search(String word) {
    Node curr = root;
    for(int i = 0;i<word.length();i++){
        char c = word.charAt(i);
        int index = c-'a';

        if(curr.children[index] == null) return false;
        curr = curr.children[index];
    }
    return curr.isWord;
}
```

startsWith

```java
public boolean startsWith(String prefix) {
    Node curr = root;
    for(int i = 0;i<prefix.length();i++){
        char c = prefix.charAt(i);
        int index = c-'a';

        if(curr.children[index] == null)
            return false;

        curr = curr.children[index];
    }
    return true;
}
```

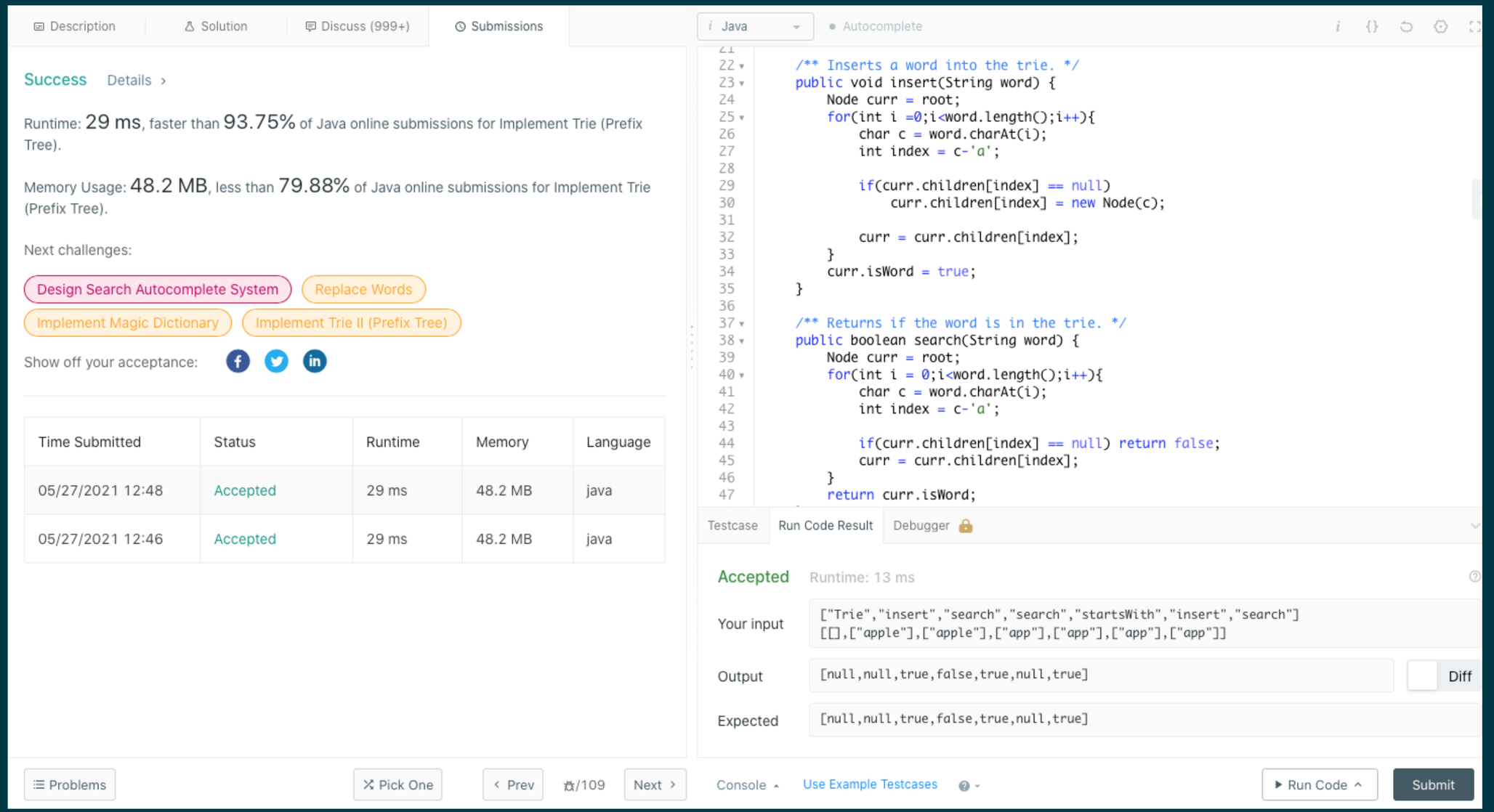
traversal

```java
public void traverse() {
    traverse(root);
}

private void traverse(Node root) {
    if(root != null) {
        System.out.println(root.value);

        for(var child: root.children) {
            traverse(child);
        }
    }
}
```

Leetcode Trie



https://leetcode.com/problems/implement-trie-prefix-tree/

Task 1
Darsda o`tilgan Trie(Array) da so`zni o`chirish imkoniyatini yarating.

public void remove(String word)

Task 2
Darsda o`tilgan Trie dagi array children o`rniga HashMap ishlatib Trie ni boshqatdan qurib chiqing.

HashMap<Character, Node> children

Task 3
HashMap orqali yaratilgan Trie ni Leetcode da tekshiring.

https://leetcode.com/problems/implement-trie-prefix-tree/