Boggle Solver

**Problem Statement**

Boggle is a word game invented by Allan Turoff. The game begins by shaking a covered tray of 16 cubic dice, each with a different letter printed on each of its sides. The dice settle into a 4×4 tray so that only the top letter of each cube is visible. The goal is to search for words that can be written using adjacent letters (i.e horizontal, vertical and diagonal manner) on the board. Based on the length of the word formed using the adjacent letters the score is alloted.

This program expects us to

**Find all Valid words on the board**

Find all the valid words that can be obtained using the letters that are available on the board.

**Calculate score for the word**

Calculate the score of the word if it is available in the dictionary that consists of all the valid words based on the length of the word.

**Related Concepts**

1. Used DFS in order to traverse the board in all possible directions, TreeSET to reduce the time complexity to find the valid words that can be possible using the letters on the board.

**Test Cases**

1. Four Test Cases are failed and scored 88/100
2. The four test cases are three memory and one Time Complexity test case

**API**

public class BoggleSolver

{

// Initializes the data structure using the given array of strings as the dictionary.

// (You can assume each word in the dictionary contains only the uppercase letters A through Z.)

**Time Complexity :** Proportional to number of words in dictionary

**Space Complexity :** Proportional to number of words in dictionary

public BoggleSolver(String[] dictionary)

// Returns the set of all valid words in the given Boggle board, as an Iterable.

**Time Complexity :** Proportional to number of rows and columns

**Space Complexity :** Proportional to number of valid words in board

public Iterable<String> getAllValidWords(BoggleBoard board)

// Returns the score of the given word if it is in the dictionary, zero otherwise.

// (You can assume the word contains only the uppercase letters A through Z.)

**Time Complexity :** Proportional to length of the word

**Space Complexity :** Constant

public int scoreOf(String word)

}