**Boggle Solver**

**Problem Statement**

Boggle Solver is used to retrieve all the valid words from a given board

**Boggle Game :**

Boggle is a word game designed by Allan Turoff and distributed by Hasbro. It involves a board made up of 16 cubic dice, where each die has a letter printed on each of its 6 sides. At the beginning of the game, the 16 dice are shaken and randomly distributed into a 4-by-4 tray, with only the top sides of the dice visible. The players compete to accumulate points by building valid words from the dice, according to these rules:

* A valid word must be composed by following a sequence of adjacent dice—two dice are adjacent if they are horizontal, vertical, or diagonal neighbors.
* A valid word can use each die at most once.
* A valid word must contain at least 3 letters.
* A valid word must be in the dictionary (which typically does not contain proper nouns).

**This program expects us to**

* Store the words in a given dictionary into a data structure (i.e., TrieST).
* Find the score for each word in the dictionary.
* Retrieve all the valid words from a given board by recursively calling the function by taking adjacent dice each time.

**Related Concepts**

* Programming Language - Java
* Two-dimensional Arrays
* TrieST
* Set
* Recursion

**Test Cases**

* All correctness test cases are passed.
* Few memory test cases are failed.
* One timing test case failed.
* The Score for the Project is 83 / 100.

**API**

**public class BoggleSolver**

**{**

**Time Complexity : Linear**

**Space Complexity : Linear**

**public BoggleSolver(String[] dictionary)**

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

**Time Complexity : proportional to m \* n**

**Space Complexity : proportional to m \* n**

**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 m(length of the word)**

**Space Complexity : Constant**

**public int scoreOf(String word)**

**}**