**Project Report – Boggle Solver**

**Problem Statement:** Boggle is a word game planned by Allan Turoff and circulated by Hasbro. It includes a barricade made of 16 cubic dice, where each bite the dust has a letter imprinted on every one of its 6 sides. Toward the start of the game, the 16 dices are shaken and arbitrarily disseminated into a 4-by-4 plate, with just the top sides of the dices noticeable. The players contend to amass focuses by building valid words from the shakers, as indicated by the following requirements.

**Requirements are:**

* A substantial word must be formed by following a grouping of adjacent dice—two dices are adjoining in the event that they are flat, vertical, or askew neighbors.
* A legitimate word can utilize each die only once.
* A substantial word must contain in any event 3 letters.
* A legitimate word must be in the lexicon (which ordinarily doesn't contain formal people names, places or things).

**Steps to do:**

1.Read the given the lexicon of words into TST in order to ration the space and time for the recovery of a specific word.

2.Recursively discover all the potential words by taking one character a period by taking neighboring dice.

Addition of the potential words into Set for putting away all potential words from the word for existence imperatives

**Related Concepts:**

* Two dimensional arrays - one for values and another for storing the Boolean values.
* TST (Ternary Symbol Table)
* Programming Language – Java

**Code:**

**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 O(l\*(n+1))

//space complexity O(l\*(n+1))

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

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

//time complexity O (h\*w\*l)

//space complexity O (h\*w\*l)

//l - length of word

//h - height of board

//w - width of 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 O (1)

//space complexity O (1)

**public int scoreOf (String word)**

**}**

**Difficulties Faced:**

Logic creation is hard to get considering the memory and time imperative.

TrieST won't fill in as the space assignment is more when contrasted with project requirements.

**Testcases:**

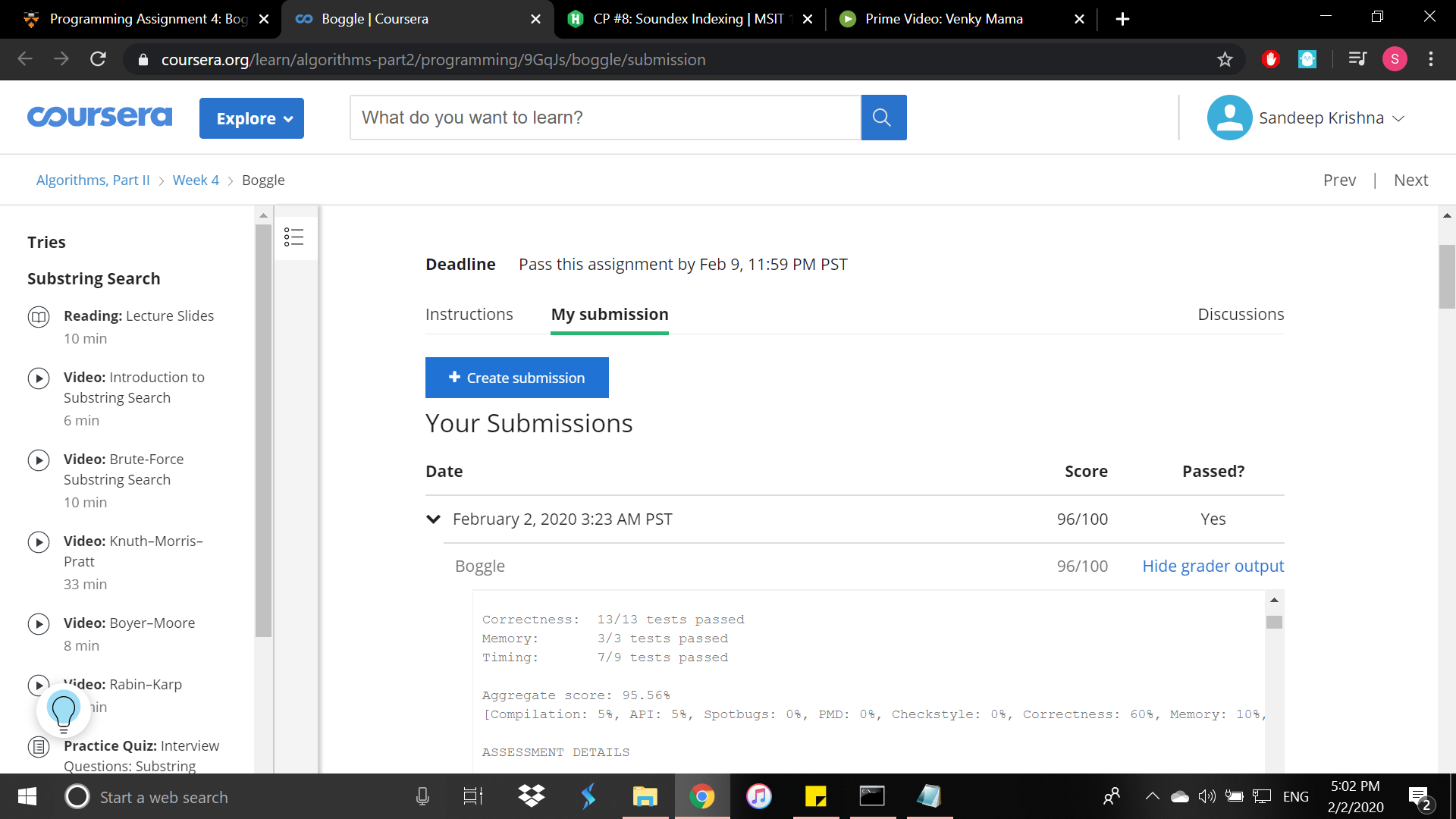
\* dictionary-zingarelli2005.txt -

student solution time (in seconds): 0.56

reference solution time (in seconds): 0.08

ratio: 6.82 ==> FAILED

**Except the above test case all other test cases were passed.**



**Sandeep Krishna**

**2019501035**