# Boggle README

Name: Nisakorn Valyasevi

NetID: nv23

Hours spent: 13 hrs

Consulted with: UTA Daniel, Chisom

Resources used: stackoverflow.com

Impressions: The instructions were more explanatory and easier to understand compared to previous assignments. It was very stressful.

Files submitted: AbstractAutoPlayer, AbstractPlayer, BadAutoPlayer, BadWordOnBoardFinder, BinarySearchLexicon, BoardCell, BoardFirstAutoPlayer, BoggleBoard, BoggleBoardFactory, BoggleGUI, BoggleMain, BoggleScore, BoggleStats, Cube, ExpandableList, GoodWordOnBoardFinder, HumanPlayer, IAutoPlayer, IBoardMaker, ILexicon, IPlayer, IPlayerView, IWordOnBoardFinder, LexiconBenchmark, LexiconFirstAutoPlayer,  LexStatus, SimpleLexicon, StandardBoardMaker, StoppableReader, TestLexicon, TestWordFinder, TrieLexicon, bogwords, kwords5, lowerwords, ospd3

## Boggle Analysis

The following is the result of running LexiconBenchmark on the different classes:

```
size of SimpleLexicon: 80612
     iter time: 0.038000   size: 80612
     word time: 0.035000   words: 80612
     pref time: 0.047000   size: 16466

size of TrieLexicon: 80612
     iter time: 1.216000   size: 80612
     word time: 0.013000   words: 80612
     pref time: 0.024000   size: 16466

size of BinarySearchLexicon: 80612
     iter time: 0.003000   size: 80612
     word time: 0.009000   words: 80612
```

```
pref time: 0.044000   size: 16466
```

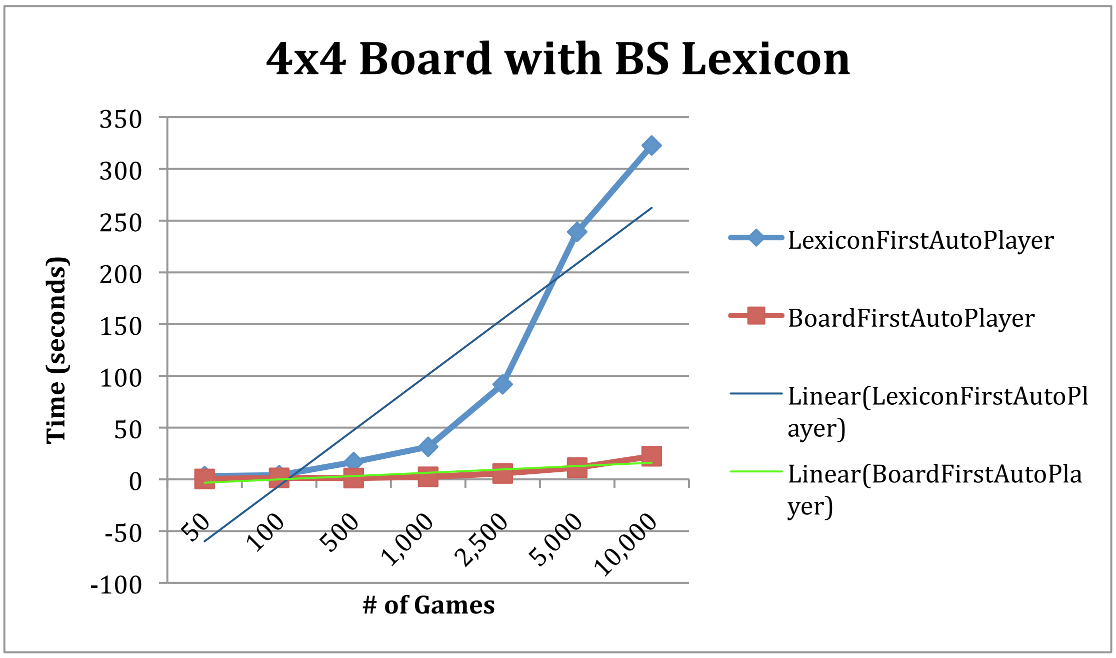
We see that Binary Search Lexicon is the fastest in terms of iteration time, the time it takes to iterate through the entire lexicon, and word time, the time it takes to check if a word is in the lexicon. However, the time it takes to identify a prefix is longer than Trie Lexicon, because the usage of tries makes finding prefixes more efficient. However, having the words organized as tries also make its iteration time much longer that other Lexicons. Since Simple Lexicon runs through every word it stored (runtime of O(N)), it takes a longer amount of time to do anything than Binary Search.

In the charts and graphs below, we see the running time of LexiconFirstAutoPlayer compared to BoardFirstAutoPlayer. We see that BoardFirstAutoPlayer runs faster which is because BoardFirstAutoPlayer searches the board, cell by cell, to see what possible words there are. However, LexiconFirstAutoPlayer goes through the all the words in the lexicon and checks if they are on the board, thus, taking much more time. We can see a linear relationship between the number of games played and the time taken, as represented in the graphs below. There are two data sets; one for a 4x4 Boggle board and the other for a 5x5 Boggle board, both tested using Binary Search Lexicon.

4x4 Board with Binary Search Lexicon

| # of Games | Time for LexiconFirstAutoPlayer (seconds) | Time for BoardFirstAutoPlayer (seconds) |
|---|---|---|
| 50 | 3.140 | 0.399 |
| 100 | 4.092 | 1.428 |
| 500 | 16.701 | 1.268 |
| 1,000 | 31.401 | 2.45 |
| 2,500 | 91.824 | 5.726 |
| 5,000 | 239.149 | 11.32 |

| | | |
|---|---|---|
| 10,000 | 322.619 | 22.341 |



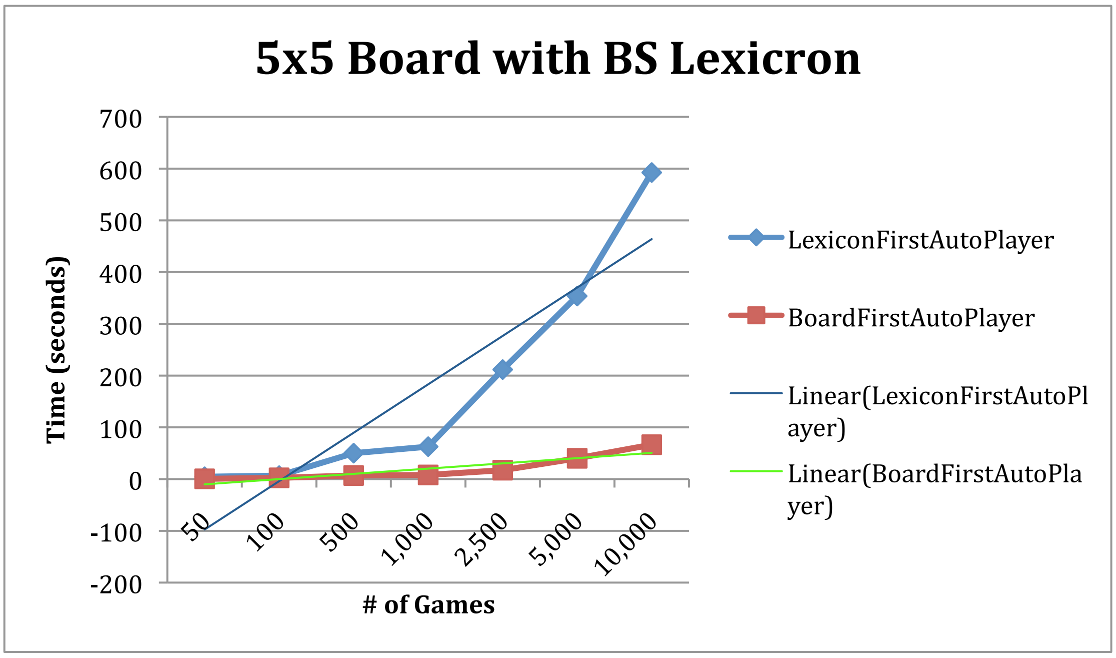
Highest Scoring 4x4 Board with Binary Search Lexicron

```
g  s  r  g
n  e  t  i
i  o  s  b
p  r  e  n

Maximum score:   889
```

5x5 Board with Binary Search Lexicon

| # of Games | Time for LexiconFirstAutoPlayer (seconds) | Time for BoardFirstAutoPlayer (seconds) |
|---|---|---|
| 50 | 4.546 | 0.508 |
| 100 | 6.698 | 2.457 |
| 500 | 50.113 | 6.950 |
| 1,000 | 62.689 | 8.017 |
| 2,500 | 211.655 | 17.246 |
| 5,000 | 353.740 | 40.174 |
| 10,000 | 592.460 | 66.450 |



Highest Scoring 5x5 Board with Binary Search Lexicron

```
  p  a  c  o  d
  o  x  s  e  r
  a  t  n  t  r
  n  i  e  a  s
  d  r  n  c  e

Maximum score:  2120
```