Boggle Analysis

Method:

In BoggleStats main function, ILexicon lex is constructed as new SimpleLexicon(),new TrieLexicon(), new BinarySearchLexicon() and new CompressedTrieLexicon() respectively to compare the performance of these lexicons. The BoggleBoardFactory class using a random number generator can generate a sequence of different Boggle boards. The setRandom method helps generate a reproducible sequence of boards, so that comparisons across different implementations of lexicons and autoplayers are valid. I supplement the code so that throughout this process of games, the board tested so far that has the highest score as well as the record score are stored. In this way, at the end of the games, the max score and the max-score board can be returned.

In LexiconBenchmark, it estimates the time its takes for each lexicon to iterate through the lexicon, find the word, and find the prefix. When evaluate TrieLexicon and CompressedTrieLexicon, lex.nodeCount() and lex.oneWayCount() is added to the code, so that it can show the node counts and oneway counts. This is useful for the trade-off analysis.

Empirical Runtime Analyses:

1) LexiconBenchmark

Three independent Benchmark test are make and the average runtimes for iter, word, pref are calculated.

Table 1	SimpleLexicon	BinarySearchLexicon	TrieLexicon	CompressedTrieLexicon
iter time: (ave of 3)	0.010333	0.009667	0.123667	0.299667
word time: (ave of 3)	0.008667	0.001000	0.000667	0.000667
pref time: (ave of 3)	0.025000	0.023333	0.021333	0.080667
node Count			153759	114921
oneWay Count			70578	31740

As is shown in **table 1**:

i) TrieLexicon and CompressedTrieLexicon take longer than binarySearch and SimpleLexicon to iterate through the lexicon.

ii) SimpleLexicon takes longest to find words.

iii) CompressedTrieLexicon takes longest to find prefix.

iv) Although CompressedTrieLexicon is slower, it has fewer node Counts and oneWay counts than TrieLexicon.

2) BoggleStats

The Output of BoggleStats is summarized in the **tables2** and **table 3**. The boards that score the highest of all the 4x4 and 5x5 boards in 1,000 games, 10,000 games and 50,000 games are also listed.

As is shown, LexiconFirstAutoPlayer is much slower than BoardFirstAutoPlayer. For this consideration, LexiconFirstAutoPlayer is only run for 1,000 and 10,000 of 4x4 and 5x5 board games. The rest of the timing for LexiconFirstAutoPlayer is predicted. Meanwhile, BoardFirstAutoPlayer is run for 1,000 games, 10,000 games and 50,000 games of 4x4 and 5x5 boards games, while the timing for 100,000 and 1,000,000 games is predicted.

These predictions are based on the theory that for a specific Lexicon implementation, the time it takes to run the code is linear correlated with the number of games. This theory is justified by the trend shown by the empirical runtimes. For example, the time it takes to run 10,000 games is roughly ten times it takes to run 1,000 games. Thus, it is predicted that running 50,000 games is five times it takes to run 10,000 games, and so on.

Conclusions

From both LexiconBenchmark and BoggleStats, it can be seen that BinarySearch Lexicon is faster than SimpleLexicon., and is the best in terms of speed.

The trade-off between speed and memory is also well demonstrated: TrieLexicon runs faster than CompressedTrieLexicon, while CompressedTrieLexicon saves more memory.

Table 2							
4x4 board		Simple Lexicon	BinarySearch Lexicon	Trie Lexicon	Compressed TrieLexicon	Max Score	Max Board
1,000 games	LexiconFirst AutoPlayer	103.415000	102.164000	165.946000	169.346000	889	g s r g n e t i i o s b p r e n
	BoardFirst AutoPlayer	1.528000	0.995000	1.623000	1.879000		
10,000 games	LexiconFirst AutoPlayer	1038.003000	1038.624000	1661.381000	1599.388000	889	g s r g n e t i i o s b p r e n
	BoardFirst AutoPlayer	11.185000	8.354000	7.158000	17.862000		
50,000 games	LexiconFirst AutoPlayer (Predicted)	~ 5190.015000	~ 8306.905000	~ 5193.120000	~ 7996.940000	1011	c l i t s m e r b d a s c l e h
	BoardFirst AutoPlayer	49.665000	30.314000	37.847000	80.118000		
100,000 games (Predicted)	LexiconFirst AutoPlayer	~10380.030000	~10386.240000	~16613.810000	~15993.880000		
	BoardFirst AutoPlayer	~111.850000	~83.540000	~71.580000	~178.620000		
1,000,000 games (Predicted)	LexiconFirst AutoPlayer	~103800.300000	~103862.400000	~166138.100000	~159938.800000		
	BoardFirst AutoPlayer	~1118.500000	~835.400000	~715.800000	~1786.200000		

Table 3							
5x5 board		Simple Lexicon	BinarySearch Lexicon	Trie Lexicon	Compressed TrieLexicon	Max Score	Max Board
1,000 games	LexiconFirst AutoPlayer	168.962000	167.546000	228.002000	227.734000	1301	o t r p w d b n o l r e s e s s t n i m w n i s h
	BoardFirst AutoPlayer	3.632000	2.318000	2.951000	5.279000		
10,000 games	LexiconFirst AutoPlayer	1706.017000	1679.376000	2292.921000	2256.152000	2120	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
	BoardFirst AutoPlayer	31.254000	21.710000	18.311000	48.112000		
50,000 games	LexiconFirst AutoPlayer (Predicted)	~ 8530.085000	~ 8396.880000	~ 11464.605000	~ 11280.760000	2120	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
	BoardFirst AutoPlayer	136.742000	106.974000	85.117000	233.513000		
100,000 games (Predicted)	LexiconFirst AutoPlayer	~ 17060.17000	~ 16793.76000	~ 22929.21000	~ 22561.52000		
	BoardFirst AutoPlayer	~ 273.484000	~ 213.948000	~ 170.234000	~ 467.026000		
1,000,000 games (Predicted)	LexiconFirst AutoPlayer	~ 170601.7000	~ 167937.6000	~ 229292.1000	~ 225615.2000		
	BoardFirst AutoPlayer	~ 2734.840000	~ 2139.480000	~ 1702.340000	~ 4670.260000		