

Algorithmics	Student information	Date	Number of session
	UO: UO269412		6
	Surname: Carrillo		
	Name: Javier		

## Activity 1. Test cases

session6.boggle.BoggleTest [Runner: JUnit 4] (542,513 s)

- testCase100 (0,122 s)
- testCase205 (0,228 s)
- testCase500 (101,585 s)
- testCase1000 (425,639 s)
- testCase01 (0,025 s)
- testCase02 (0,029 s)
- testCase03 (0,040 s)
- testCase04 (0,060 s)
- testCase05 (0,013 s)
- testCase15 (0,035 s)
- testCase46 (0,404 s)
- testCase205\_2 (14,333 s)

### Failure Trace

java.lang.AssertionError: expected:<957> but was:<955>  
at session6.boggle.BoggleTest.testCase15(BoggleTest.java:97)

The test case 15 probably won't work due to the QS that exists in table, but I thought the implementation I did supported more than one letter in the board (subtracting the length of the word).

**UPDATE:** After changing the file of the table 15, the test now succeeds.

Runs: 1/1 Errors: 0 Failures: 0

testCase15 [Runner: JUnit 4] (0,177 s)

Algorithmics	Student information	Date	Number of session
	UO: UO269412		6
	Surname: Carrillo		
	Name: Javier		

## Activity 2. Times for different executions

n	Time (ms)
1	1
11	14
21	38
31	177
41	324
51	419
61	807
71	1233
81	2079
91	2841
101	3552
111	5611
121	6485
131	7247
141	7166
151	7961
161	8905
171	10862
181	11931
191	12581
201	13974
211	14941
221	15760
231	18127
241	20203
251	21739
261	22934
271	26570
281	28403
291	31014
301	32708
311	36493
321	39354
331	39470
341	43131
351	46380

Algorithmics	Student information	Date	Number of session
	UO: UO269412		6
	Surname: Carrillo		
	Name: Javier		

361	48507
371	52903
381	55728
391	59437
401	61236
411	64851
421	68463
431	76052
441	77368
451	82050
461	85080
471	87219
481	92181
491	95852
501	102314

The complexity of the problem follows a linear complexity, as can be seen taking some theoretical values (theoretical value of  $n = 401 \rightarrow 60957$ , close to the 61236 obtained. Theoretical of  $n = 261 \rightarrow 22605$ , close to the 22934 obtained.)

## Activity 3. Code improvements

When first approaching the problem, I used an array list to avoid any kind of complications regarding the structure that stores the dictionary and focus on the backtracking implementation. After finally passing some tests, I realized that this structure was far from optimal so I exchanged it for an Hashtable of strings. Realizing it was still slow, I implemented an Hashtable of strings and booleans, the booleans indicating if that word in the dictionary was already in the solutions. Realizing it was still small, I thought about dividing the Hashtable into different hashtables that would store sections of the dictionary (one from a to m, other to m to r and so), thus converting the structure into a `hashtable<hashtable<string,boolean>, string>`. However, while looking for ways to implement a better dictionary, I found the prefix tree structure (which used a far more

Algorithmics	Student information	Date	Number of session
	UO: UO269412		6
	Surname: Carrillo		
	Name: Javier		

developed “sectioning” than the one that I planned to, so I implemented that structure in the end).