





# PRÁCTICA 6

Ángela López López UO270318

## ACTIVIDAD 1 . CASOS DE PRUEBA

**PC CASA:** Intel ® Core(TM) i5-8250U CPU @ 1.60Ghz 1.80 GHz, MEMORIA: 8.00 GB

- ✓  BoggleTest [Runner: JUnit 5] (29,981 s)
  - ✓  testCase100 (0,037 s)
  - ✓  testCase205 (0,085 s)
  - ✓  testCase500 (5,489 s)
  - ✓  testCase1000 (23,323 s)
  - ✓  testCase01 (0,018 s)
  - ✓  testCase02 (0,012 s)
  - ✓  testCase03 (0,018 s)
  - ✓  testCase04 (0,010 s)
  - ✓  testCase05 (0,013 s)
  - ✓  testCase15 (0,018 s)
  - ✓  testCase46 (0,061 s)
  - ✓  testCase205 2 (0,897 s)

## ACTIVIDAD 2. TIEMPO PARA LAS DIFERENTES EJECUCIONES

Tamaño: 1 - Tiempo: 38 ms - Puntos: 0  
Tamaño: 2 - Tiempo: 31 ms - Puntos: 13  
Tamaño: 3 - Tiempo: 16 ms - Puntos: 123  
Tamaño: 4 - Tiempo: 16 ms - Puntos: 155  
Tamaño: 5 - Tiempo: 0 ms - Puntos: 113  
Tamaño: 6 - Tiempo: 15 ms - Puntos: 389  
Tamaño: 7 - Tiempo: 0 ms - Puntos: 428  
Tamaño: 8 - Tiempo: 0 ms - Puntos: 1095  
Tamaño: 9 - Tiempo: 16 ms - Puntos: 1405  
Tamaño: 10 - Tiempo: 16 ms - Puntos: 923  
Tamaño: 11 - Tiempo: 6 ms - Puntos: 1934  
Tamaño: 12 - Tiempo: 15 ms - Puntos: 2220  
Tamaño: 13 - Tiempo: 32 ms - Puntos: 4861  
Tamaño: 14 - Tiempo: 15 ms - Puntos: 3581  
Tamaño: 15 - Tiempo: 22 ms - Puntos: 7933

Tamaño: 16 - Tiempo: 16 ms - Puntos: 3860  
Tamaño: 17 - Tiempo: 16 ms - Puntos: 7671  
Tamaño: 18 - Tiempo: 33 ms - Puntos: 9438  
Tamaño: 19 - Tiempo: 20 ms - Puntos: 6798  
Tamaño: 20 - Tiempo: 18 ms - Puntos: 8809  
Tamaño: 21 - Tiempo: 19 ms - Puntos: 9121  
Tamaño: 22 - Tiempo: 18 ms - Puntos: 12186  
Tamaño: 23 - Tiempo: 21 ms - Puntos: 9326  
Tamaño: 24 - Tiempo: 20 ms - Puntos: 12611  
Tamaño: 25 - Tiempo: 28 ms - Puntos: 14117  
Tamaño: 26 - Tiempo: 20 ms - Puntos: 11131  
Tamaño: 27 - Tiempo: 20 ms - Puntos: 13874  
Tamaño: 28 - Tiempo: 23 ms - Puntos: 17316  
Tamaño: 29 - Tiempo: 20 ms - Puntos: 16993  
Tamaño: 30 - Tiempo: 23 ms - Puntos: 22101

### ¿Qué piensas de la complejidad del algoritmo?

La complejidad de este algoritmo es  $O(8^{n^2})$  porque cada nodo tiene como mucho 8 vecinos directos.

En el juego las palabras se pueden formar de manera arbitraria (ya que se puede seleccionar cualquier carácter que está adyacentemente situado y no se haya utilizado previamente). Por lo tanto para una palabra de longitud  $L$ , existen  $8^L$  combinaciones posibles. Consideramos  $L$  como constante (ya que el diccionario tiene una combinación constante de caracteres, 26). Por lo tanto encontrar palabras partiendo de una posición dada tiene una complejidad  $O(1)$ .

Sin embargo también hay que tener en cuenta la posición inicial de la palabra, que es  $n^2$  por lo que la complejidad total es  $O(8^{n^2})$

## ACTIVIDAD 3. MEJORAS EN EL CÓDIGO

**Dependiendo de factores como las estructuras de datos utilizadas y la posibilidad de podar el árbol el rendimiento del algoritmo puede cambiar drásticamente. ¿Has hecho alguna elección que creas que mejora el rendimiento de tu algoritmo? ¿Cual?**

Después de muchas horas de investigación sobre estructuras de datos que se pudiesen aplicar a este problema decidí utilizar unas cuantas.

Para empezar guardo las palabras encontradas en el juego en un TreeSet. Esta estructura no permite duplicados y ordena los datos alfabéticamente. Es muy útil ya que deben estar ordenadas para poder comprobar los resultados en los tests. Si no tuviesen que estar ordenadas HashSet sería mejor opción ya que es más eficiente.

Para ordenar las palabras del diccionario he decidido utilizar un Trie. Se trata de una estructura de tipo árbol de búsqueda que usa letras como claves. Un nodo no almacena la clave asociada con ese nodo sino que su posición indica con qué nodos está asociado. Es decir, los nodos descendientes tienen en común el mismo prefijo, y la raíz está asociada con un string vacío.

Este tipo de estructura se utiliza para formar palabras, ya que cada nodo es un carácter.

Si el tablero solo tiene 1 carácter, no se pueden formar palabras válidas por lo que no es necesario comprobar estos casos. De la misma forma una palabra solo puntúa si tiene al menos 2 caracteres.

Para tratar el backtracking del grafo usamos el camino en profundidad, usando una matriz auxiliar de visitados que nos permita no procesar el mismo nodo varias veces.