Universidad Simón Bolívar
CI-3661 - Lab Lenguajes de programación
Trimestre Abril-Julio 2022

Eros Cedeño 16-10216 Prof Fernado Torres Mora

# Proyecto Haskell

(Wordle Game)

## Introduccion

Wordle es un juego basado en el popular juego de Toros y Vacas donde una mente maestra piensa en un palabra (dentro de una lista de palabras) y un descifrador va tanteando con palabras (también dentro de la lista) y recibiendo retroalimentación de esta, hasta adivinar la palabra o hasta que se cumplan los 6 turnos permitidos.

El presente informe consta de cuatro secciones en las que se exploran las decisiones de diseño del proyecto, sus dificultades de implementación, las optimizaciones que se incorporaron, las conclusiones y recomendaciones para quienes pudieran realizar el proyecto en el futuro.

## Decisiones de Diseño

Se diseñó el código como dos módulos de juego y un módulo cliente de ejecución. El módulo mentemaestra, el módulo descifrador y el módulo wordle respectivamente. Esto para conservar la independencia de ambos modos de juego y que se pudieran compilar desde diferentes archivos sin problemas de dependencias. Se buscó explotar al máximo el excelente desempeño que posee Haskell sobre el manejo de

Listas. La gran eficiencia de las funciones Map, fold y zip sostienen en gran medida el razonamiento de los problemas.

En el módulo de mentemaestra se implementó la interacción con el usuario a través de monats. Y en el módulo Descifrador se busco reducir al mínimo el uso de monats para asegurar la pureza de las funciones y aprovechar la seguridad que proporciona la ausencia de efectos de borde en Haskell.

En el módulo Descifrador se implementó el árbol mediante listas, esto para aprovechar la versatilidad de Haskell para operar sobre dicha estructura. Sobre estas listas de palabras se usó la técnica minimaxing para asegurar predicciones de palabras más cercanas a razonamientos humanos y que se acerque inteligentemente a la respuesta correcta.

En este módulo también se toma en consideración la posibilidad de que el jugador humano esté intentando hacer trampa despistando a la computadora o dando pistas contradictorias. Esto para hacer el programa más robusto y estable.

En el módulo mentemaestra el input del usuario se restringe a palabras posibles de la lista de palabras. Las palabras introducidas por el usuario son verificadas antes de proceder para asegurar que su respuesta cuente como un intento valido. Si el input recibido no pertenece a la lista de palabras se informará al usuario de su error y no se tomará en cuenta dicho intento.

En ambos módulos se deja el nombre del archivo a consultar en una constante global llamada archivo. En caso de querer cambiar el archivo de origen solo es necesario modificar la ruta en estas constantes.

Como optimización de ejecución del modo Descifrador, al inicio se genera el árbol calificado mediante listas y se elige la palabra con menor calificación. Luego, por cada iteración o intento de hallar la palabra del usuario se poda el árbol dependiendo de la respuesta recibida. Dicha respuesta se califica y en base a esta se decide.

**Dificultades de Implementación**

Al implementar el modo MenteMaestra se presentaron algunas dificultades. La evaluación perezosa en ocasiones genera desfases al realizar interacciones con el usuario (Std I/O) dado que algunas sentencias o instrucciones eran saltadas o ejecutadas desordenadas por lo que se usa la directiva ¨hFlush stdout¨ para asegurar la ejecución de algunas instrucciones de interacción.

El modo Descifrador fue el más desafiante. La manipulación eficiente del árbol fue un punto fundamental. Generar una estructura de datos recursiva que represente un árbol 10-ario con profundidad 4 por cada iteración de este modo era algo inaudito, se soluciono este problema representando esta estructura sobre listas y así aprovechamos el amplio repertorio de funciones sobre listas nativas en haskell para realizar los cálculos necesarios.

Se tuvo dificultades al diseñar la adivinación correcta de palabras con letras repetidas. Como por ejemplo ¨ERRAR¨ o ¨ABABA¨con tres caracteres repetidos o ¨TABACO¨ y ¨EMANA¨ con dos caracteres repetidos. Sin embargo se logró solucionar y optimizar dichas adivinaciones estableciendo un proceso progresivo de poda de las posibles respuestas, de tal manera que cada pista obtenida del usuario permite verificar si un nodo cumple con las condiciones hasta ahora establecidas. Y si en algún instante una letra es indicada como ¨-¨ entonces se

verifica cuidadosamente el uso de dicha letra para evitar podas no intencionadas que puedan imposibilitar hallar la palabra buscada.

**Optimizaciónes**

Como ya se ha mencionado el uso de listas para la manipulación de la data en el modo descifrador fue una excelente decisión de diseño dado que muchas de las operaciones sobre listas en haskell son a lo sumo O(n) y otras como sort están implementadas con alto rendimiento y flexibilidad aprovechando los núcleos del procesador para ejecutar cálculos en paralelo.

**Conclusiones y Recomendaciones**

El proyecto fue muy entretenido y divertido. Permite crecer y aprender un lenguaje completamente nuevo y además funcional como es Haskell. Es un lenguaje con un poder expresivo muy grande y muy similar a las estructuras formales logico-matemáticas que tanto nos gustan.

Se recomienda que si se va a realizar este proyecto a futuro, se ejercite en el uso de los cuantificadores, de las listas por comprensión y las funciones map, fold, zip, unzip, head, tail, sort entre otros. También se recomienda compilar el código con ghc --make wordle. Es recomendable entender a profundidad el problema antes de aventurarse a programar. Se recomienda tener clara una estructura del código, un criterio claro de la solución a implementar, de este modo se evitan enfrentamientos continuos con los errores del compilador para luego percatarse de errores de diseño y tener que refactorizar.

Sería muy interesante que por la línea de comandos se pudiera especificar la ruta del archivo del cual se toman las

palabras del juego. Además se recomienda poder enviar otro parámetro N tal que N sea la cantidad de intentos permitidos por partida, de este modo el juego puede extenderse a gusto del jugador por más de 6 iteraciones por partida.

*Las grandes obras son hechas no con la fuerza, sino con la perseverancia.*