ESTRUCTURAS DE DATOS Y ALGORITMOS

TRABAJO PRÁCTICO ESPECIAL

ALUMNOS

- 52056 – Juan Marcos Bellini
- 55291 – Julián Rodriguez Nicastro
- 55824 – Juan Li Puma

ITBA 55 años
INSTITUTO TECNOLOGICO DE BUENOS AIRES
•UNIVERSIDAD PRIVADA•

TABLA DE CONTENIDOS

INTRODUCCIÓN

El presente informe trata acerca del trabajo práctico especial de la materia Estructuras de Datos y Algoritmos. El mismo consiste en la implementación de un algoritmo que resuelve tableros de una variante del juego *Scrabble*.

Para poder cumplir con el objetivo, se han desarrollado dos algoritmos. Por un lado, uno encuentra el tablero que más puntos sume, en el tiempo que sea necesario. Por otro lado, el segundo obtiene la mejor solución hasta el momento (indicándole durante cuánto tiempo tiene que ejecutarse).

A continuación se describen ambos algoritmos, las estructuras de datos utilizadas, los problemas encontrados durante el desarrollo, y las decisiones tomadas para abordarlos, y comparaciones entre ambos algoritmos.

ALGORITMOS UTILIZADOS

Para poder encontrar soluciones al problemas, se implementaron dos algoritmos, uno exacto y otro aproximado. El exacto, como lo indica su nombre, encuentra la mejor solución al problema. Esto es, el tablero con mayor puntaje. En cambio, el segundo, encuentra una solución que puede, o no, ser la mejor, pero que es aceptable.

El problema del primer algoritmo es que es demasiado complejo. Al tratar de buscar la mejor solución, se debe realizar numerosas pruebas, hasta llegar al tablero buscado. Si la cantidad de letras y palabras disponibles es muy grande, el algoritmo podría tardar años, o incluso siglos. Por lo tanto, el hecho de poder obtener una solución aceptable – mediante un algoritmo de aproximación – permite resolver problemas que, de otra forma, si bien se resuelve de la manera más óptima, habría que esperar mucho tiempo hasta lograr dar con dicha solución.

ALGORITMO EXACTO

Para obtener la mejor solución posible al problema planteado, se decidió utilizar *backtracking.* Esta técnica consiste en recorrer en profundidad un árbol de soluciones parciales. Al llegar a una hoja de dicho árbol (estado a partir del cual no se puede obtener más soluciones), si dicha solución es mejor que la mejor obtenida hasta el momento, se la remplaza; si no es mejor, se la descarta. El algoritmo termina cuando se recorra todo el árbol o, en el caso a analizar, cuando se obtenga un tablero en el que se hayan ubicado todas las letras.

A continuación se describe el algoritmo en pseudo-código.

```
Sea L el conjunto de Letras que se tiene
Sea D el conjunto de Palabras // El diccionario
Sea mejorSolución un Estado con Tablero vacío // Guardará la mejor solución
Sea máximoPuntaje = Suma(Puntajes de Letras en L)

FUNCIÓN backTrackingSolve(Estado e)

        Sea M el conjunto de Movimientos posibles a partir del Estado e

        SI M es vacío
        ENTONCES:
                SI e.Puntaje > mejorSolucion.Puntaje
                ENTONCES:
                        mejorSolucion = e
                        SI mejorSolución.Puntaje == máximoPuntaje
                        ENTONCES:
                                Terminar la ejecución
                        FIN SI.
                FIN SI.
        SINO:
                PARA TODO movimiento en M:
                        Aplicar movimiento a e
                        backTrackingSolve(e)
                        Quitar movimiento de e
                FIN PARA TODO
        FIN SI.
FIN FUNCIÓN.
```

ALGORITMO APROXIMADO

```
Sea e un Estado con Tablero vacío
Sea I el conjunto de Movimientos posibles a partir de e // Estados iniciales
Sea mejorSolución un Estado con Tablero vacío // Guardará la mejor solución
Sea máximoPuntaje = Suma(Puntajes de Letras en L)

FUNCIÓN stochasticHillClimbingSolve(Tiempo t_max)

        Sea t_inicial = 0

        PARA TODO movimiento en I
                Aplicar movimiento a e
                stochasticHillClimbingRecursive(e, t_inicial, t_max)
                SI mejorSolución.Puntaje == máximoPuntaje
                ENTONCES:
                        Terminar la ejecución
                FIN SI.
                Quitar movimiento a e
        FIN PARA TODO
FIN FUNCIÓN


FUNCIÓN stochasticHillClimbingRecursive(Estado e, Tiempo t_actual, Tiempo t_max)

        SI t_actual >= t_max
                Terminar la ejecución
        FIN SI

        SI e.Puntaje > mejorSolución
        ENTONCES:
                mejorSolución = e
                SI mejorSolución.Puntaje == máximoPuntaje
                ENTONCES:
                        Terminar la ejecución
                FIN SI.
        SINO:
                Sea M el conjunto de Movimientos posibles a partir del Estado e
                PARA TODO movimiento en M
                        Sea r un número real al azar entre 0 y 1
                        SI r < movimiento.probabilidadDeSerElegido
                        ENTONCES:
                                Aplicar movimiento a e
                                t_actual = obtenerTiempoDeEjecución
                                stochasticHillClimbingRecursive(e, t_actual, t_max)
                        FIN SI.
                FIN PARA TODO
        FIN SI.
FIN FUNCIÓN.
```