Proyecto Programado: Juego del Katamino - Paolo Induni

1. Resumen Ejecutivo

El Katamino es un juego que contiene varias piezas de diferentes formas semejantes a las de Tetris que se intentan acomodar de manera correcta en un tablero de tamaño n x m. Este juego de mesa tiene la peculiaridad que su dificultad aumenta según el espacio que se tenga que completar, el cual puede expandirse para una resolución más ardua. Un pasatiempo que combina las mátematicas con la capacidad del individuo para observar y el desarrollo de su pensamiento para completar los desafíos planteados.

Se busca resolver el Katamino con la ayuda del uso de pila, ciclos, listas y matrices para asi plantear una solución al Katamino. La solución debe ser la correcta colocación de todas las piezas en el tablero n x m, la complicación que trae este problema es que para encontrar su solución primero se debera probar varias combinaciones antes de encontrar la correcta. Asimismo es importante resaltar que las piezas pueden ser rotadas y no es necesario utilizarlas en la dirección inicial.

El Katamino se resolverá con la rotación de piezas en el juego hasta que se encuentre una manera en la que todas las piezas encajen de manera correcta en el tablero. Esto se realizará de manera que las piezas sean leídas como valores de entrada y que en forma de pila se vayan probando en la matriz, en el caso que no sea posible colocarlo se continuara con la próxima pieza hasta encontrar una solución posible para el Katamino.

Para comprobar que la solución es acertada se realizaran diversas pruebas del programa para asi evaluar si los resultados suministrados por el programa son correctos. Ádemas no solo se busca que la solución sea correcta sino tambien que la misma sea eficiente y sin demora dentro del marco de dificultad que se llegue a plantear para dicha prueba. Finalmente se procederá a comparar dichas pruebas con dificultades similares para asi evaluar las posibles soluciones con sus respectivos tiempos. En el caso que una prueba no entregue una respuesta despues de unas horas se tomara como que el programa no es capaz de resolverlo.

Los resultados encontrados demostraron que en las pruebas a mayor dificultad, mayor sería el tiempo de ejecución del programa por lo que mientras mas rotaciones mayor sería el tiempo y cuantas veces se repitiera este proceso significaria mas espera.

Despues de analizar estos resultados se llega a pensar que el programa tarda mas según las piezas que tenga que rotar y la cantidad que tenga que repetir este proceso. Cuando la dificultad aumenta el tiempo de ejecución del programa se extiende. Por lo que en aquellas pruebas de 5x5 tenían menor duración a las pruebas de 5x8 que eran de mayor dificultad. Además se tiene en cuenta que para optimizar el código queda pendiente una forma en la que el se descarten arboles que no se van a utilizar para que así de esta forma se descarten casos que no son posibles

Se llega a la conclusión que el programa resuelve los Kataminos de manera

mas eficiente los de menor dificultad y con menor eficiencia los de mayor dificultad, aunque existieron casos en los que no resolvía algunas pruebas por la gran cantidad de rotaciones que implicaban las piezas en el tablero de juego.

2. Introducción

El ser humano desde tiempos antiguos ha disfrutado de los acertijos. Durante siglos la sociedad disfrutó de estos juegos de mesa. En los ultimos 70 años y con la ayuda del desarrollo de la computación, se ha logrado resolver muchos de estos juegos de tablero en tiempos que antes eran imposibles. En este caso se resolvera el Katamino.

El Katamino es un juego que contiene varias piezas de diferentes formas semejantes a las de Tetris que se intentan acomodar de manera correcta en un tablero de tamaño n x m. Este juego de mesa tiene la peculiaridad que su dificultad aumenta según el espacio que se tenga que completar, el cual puede expandirse para una resolución más ardua. Un pasatiempo que combina las mátematicas con la capacidad del individuo para observar y el desarrollo de su pensamiento para completar los desafíos planteados.

Se busca resolver el Katamino con la ayuda del uso de pila, ciclos, listas y matrices para asi plantear una solución al Katamino. La solución debe ser la correcta colocación de todas las piezas en el tablero n x m, la complicación que trae este problema es que para encontrar su solución primero se debera probar varias combinaciones antes de encontrar la correcta. Asimismo es importante resaltar que las piezas pueden ser rotadas y no es necesario utilizarlas en la dirección inicial.

Para comprobar que la solución es acertada se realizaran diversas pruebas del programa para asi evaluar si los resultados suministrados por el programa son correctos. Ádemas no solo se busca que la solución sea correcta sino tambien que la misma sea eficiente y sin demora dentro del marco de dificultad que se llegue a plantear para dicha prueba. Finalmente se procederá a comparar dichas pruebas con dificultades similares para asi evaluar las posibles soluciones con sus respectivos tiempos.

Índice

3. Marco teórico

Python es un lenguaje de programación de alto nivel. Proporciona mayor productividad debido a que no hay un paso de compilación, el ciclo de edición, prueba y depuración se convierte en algo increíblemente eficiente. La depuración de programas en Python es simple, un error o una entrada que sea incorrecta no provocará una falla de segmentación. Pero si un intérprete descubre un error, genera una excepción. Un depurador a nivel del código fuente ayuda a inspeccionar las variables locales y globales, la evaluación de expresiones arbitrarias, la designación de puntos de interrupción y el paso del código en el que se analice línea por línea.

Ádemas es un lenguaje de código abierto creado por Guido van Rossum a finales de los 80 mientras trabajaba en el sistema operativo Amoeba. Su versión 2.0 se lanza en octubre del año 2000, la cual contaba con algunas características nuevas para los lenguajes de su tipo, aunque esto no sería lo que impulsaría su avance para que llegara a ser el lenguaje que conocemos hoy en día, ya que este comenzó a ser desarrollado por la comunidad, abriendo paso a la versión 3.0. Esta versión es muy diferente a las anteriores lanzadas 8 años antes, aunque, sus desarrolladores han realizado un gran esfuerzo para compatibilizar varias características con la versión 2.6 de python.

Este lenguaje, a pesar de ser "sencillo", es bastante completo y cuenta con muchas herramientas para ayudar al usuario, lo que ha vuelto a Python uno de los favoritos en la programación moderna, haciendo que aún más personas se sumen al desarrollo de bibliotecas que faciliten procesos y aumenten la eficacia del mismo.

PyCharm por su parte es un IDE (entorno de desarrollo integrado) de Python dedicado que brinda una extensa cantidad de herramientas para los desarrolladores de Python, especialmente incluidos en el desarrollo productivo de lo que es conocido como la ciencia de datos, web y el mismo Python. En este caso, Pycharm es el entorno de programación utilizado para este proyecto, es una herramienta que permite al usuario trabajar con Python 3.0 e implementar tanto librerías como herramientas que facilitan el trabajo a la hora de programar, es uno de los IDEs más recomendados para windows, ya que logra aprovechar de la mayoría de recursos que python tiene a su disposición, sin llegar a ser demasiado complejo o consumir recursos excesivos. Sys: se utilizó para establecer un limite para las recursiones del código.

4. Descripción de la solución

La solución que se desarrolló para este proyecto fue inicialmente pensada en distintas funciones que juntas pudieran satisfacer las necesidades del problema planteado. Primeramente se crea una matriz vacía con las medidas proporcionadas por el usuario, de esta forma se podra tener el tablero del Katamino para posteriormente incluir las piezas. Una de las funciones del programa es la que está a cargo de las rotaciones, esta función ayuda a completar el tablero en una posición en la que encaje, además de asi poder acoplarse en otras direcciones con las demas piezas.

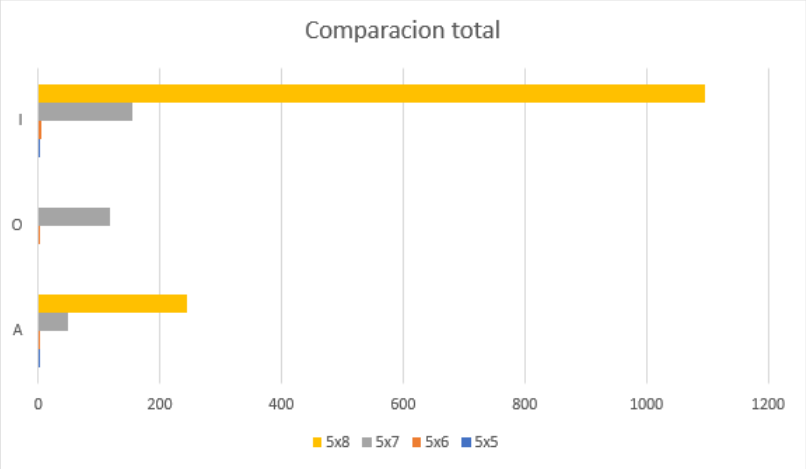
Otra función destacable es la que está a cargo de leer las piezas ingresadas las cuales posteriormente en otra función serán separadas de los espacios vacíos para así poder ser ingresadas en el tablero del juego. Hay consideraciones que se tienen que tener en cuenta a la hora de trabajar con estas piezas, un asunto que debe pensarse es si la pieza es simétrica, esto es destacable porque significaría que la pieza no tiene rotaciones posibles.

El programa presenta una solución mas rápida cuando su problema necesita menor cantidad de rotaciones, esto se debe a que el cálculo de rotaciones y si las piezas encajan juntas en esa dirección es lo que toma mas tiempo del programa, las demás funciones presentan tiempos similares. Para este proceso adicionalmente del uso de matrices, listas y pilas se utilizó la recursividad para las piezas y que asi este proceso fuera mas sencillo de programar, la recursividad se encarga de eliminar la primera pieza y continuar con aquellas piezas que sobran.

Para intentra encajar las piezas cada una de ellas se intentará colocar en un espacio que resulte estar vacío, si la pieza no encaja en ese espacio y sus rotaciones se agotan se procede a intentar con las mismas rotaciones en otra posición en la cual se comprueba si es posible intentarlo, en el caso que resulte que no se puede poner la pieza no continuara con su verificación en esa posición para así ahorrar tiempo y que sea posible verificar si se puede colocar en distintas posiciones.

En el caso que no sea posible colocar una pieza en el tablero aunque se hayan probado todas las posibles posiciones y las potenciales rotaciones se guardará en la lista y se procederá a continuar con la siguiente pieza, esto con la ayuda de la recursividad mencionada anteriormente. Aunque las rotaciones sean lo que puede llegar a tomar mas tiempo, este proceso tambien se puede extender si no se encuentra una combinación en la que todas las piezas encajen.

5. Resultados de las pruebas



Pruebas	A	O	I
5x5	0.31927514	X	0.17765284
5x6	3.59158278	0.44338083	5.83501101
5x7	48.1293325	118.722456	155.834327
5x8	245.189564	X	1096.64077

Para comparar la funcionalidad del programa se realizaron tres pruebas usando los diferentes tamaños proporcionados en el documento. Los resultados fueron:

En la prueba A fue posible calcular todos los Kataminos de diferentes dimensiones. Prueba A (5x5) 0.3192751407623291 (5x6) 3.591582775115967 (5x7) 48.129332542419434 (5x8) 245.18956398963928. En la prueba 0 se calculo el 5x6 y 5x7. Prueba O e (5x6) 0.44338083267211914 (5x7) 118.722456455230714. En la prueba I fue posible calcular todos los Kataminos de diferentes dimensiones. Prueba I (5x5) 0.17765283584594727 (5x6) 5.835011005401611 (5x7) 155.83432745933533 (5x8) 1096.6407749652863.

Despues de analizar estos resultados se llega a pensar que el programa tarda mas según las piezas que tenga que rotar y la cantidad que tenga que repetir este proceso. Cuando la dificultad aumenta el tiempo de ejecución del programa se extiende. Por lo que en aquellas pruebas de 5x5 tenían menor duración a las pruebas de 5x8 que eran de mayor dificultad. Todo esto se llega a comprender cuando se comparan los tiempos de las pruebas en los que queda en evidencia que a mayor tablero, mayor será el tiempo en el que se ejecute la solución del código.

6. Conclusiones

Se llegó a la conclusión para todo Katamino que tenga la posibilidad de ser completado, se encuentra una respuesta ya que prueba absolutamente todas las combinaciones posibles del caso específico, por lo cual si no encuentra una solución posible, significa que es porque realmente no existe una. Además se tiene en cuenta que para optimizar el código queda pendiente una forma en la que el se descarten arboles que no se van a utilizar para que así de esta forma se descarten casos que no son posibles.

Relacionado a lo anterior también se reconoce que mientras menos rotaciones tenga que realizar el programa, con mayor velocidad resolverá el problema del tablero de Katamino. Por lo que siempre que sean pocas rotaciones en las piezas y se encuentren combinaciones posibles entre las piezas el programa será fluido, pero cuando existan muchas rotaciones y las piezas no encajen a la primera el programa tendrá mas dificultad en el tiempo de ejecución.

Por lo que lo único que puede ayudar a mejorar la eficiencia para resolver el tablero sería una función que descarte casos que no son posibles para que de esta manera el tablero pueda ser resuelto de manera más rapida. Además una función que descarte las piezas que no entran en el tablero y que continue con la siguiente pieza sin tener que intentar con la pieza que no encaja para así mejorar la eficiencia del código.

Finalmente se tiene en cuenta que el código programado de otra manera mejoraría su funcionamiento. Aunque el programa final cumple con las pruebas en el tiempo esperado, se entiende que se puede mejorar la forma en la que se prueban las piezas y la manera en la que se descarta. Esto disminuiría el tiempo en el que el programa resuelve el tablero de juego.

7. Bibliografía

[1] I. Challenger-Pérez, Y. Díaz-Ricardo and R.A. Becerra-García, “El lenguaje de programación Python”, Ciencias Holguín, vol. 20, no. 2, pp. 1-13.

[2] Ramirez, E.(2017). Fundamentos de Programación. Sin editorial

[3] Dana S. Scott (1958). ”Programming a combinatorial puzzle”. Technical Report No. 1, Department of Electrical Engineering, Princeton University.