TEC | Tecnológico de Costa Rica

Estudiantes: Ana Belén Murillo, Josué Quintana, Angie Esquivel López

1 de junio

Tecnológico de Costa Rica

Introducción a la programación

Rompecabezas Deslizantes

Eddy Ramírez Jiménez

Proyecto 3 de Taller

1. Resumen ejecutivo

- **Resumen del proyecto:**
  Este proyecto trata sobre un rompecabezas deslizante, en donde se debe de implementar el algoritmo de programación el cual lleva el nombre de BFS (Breadth First Search).Dada una configuración específica, se debe resolver el rompecabezas deslizante siguiendo la regla de que solo se pueden "deslizar"las piezas, sin permitir cambios de piezas de la nada. Se busca que el código sea el más eficiente posible, por lo que se implementó varias técnicas para que esto se lograse, además se busca que los alumnos sean capaces de implementar una interfaz gráfica de manera correcta.

- **Nuestra solución:**
  El primer paso que se realizó fue el comprender y analizar el problema, ya que así se puede deducir qué pasos seguir para una implementación exitosa. Por lo que se estuvo jugando en los teléfonos celulares un videojuego de rompecabezas deslizantes.
  En términos generales, para la solución de este problema se implementaron 5 funciones. Primero se registra una matriz que representa un rompecabezas y obtiene las coordenadas del espacio vacío, que está representado por un "0", luego se crea una lista en la cual se almacena los movimientos realizados en el rompecabezas. Lo siguiente es saber las coordenadas del espacio vacío para generar una lista de posibles movimientos adyacentes, siempre y cuando estén dentro de los límites válidos del rompecabezas.Si lo están, crea una copia del rompecabezas actual, intercambia el espacio vacío con el número en la posición correspondiente, y agrega la nueva configuración a la lista de movimientos.
  Luego, para poder lograr uno de los objetivos del proyecto, la eficiencia, se llevaron a cabo varios códigos que hacían que esto fuera posible, que son los siguientes:
  La función **heu_dist** calcula la **distancia** estimada entre la configuración actual y el estado objetivo del rompecabezas.
  La función **BFS** utiliza la **búsqueda por anchura** para encontrar la solución del rompecabezas, utilizando la heurística de distancia estimada.
  Para finalizar, a lo que se refiere a interfaz gráfica esta se realizó usando Pygame, en donde se encarga de dibujar el rompecabezas en una ventana, la solución la obtiene llamando a la función **BFS** y, si hay una solución válida, se dibujan todas las configuraciones de rompecabezas necesarias para alcanzar la meta.

- **Resumen de los resultados de las pruebas:**
  Los resultados que se obtuvieron en este proyecto fueron muy favorables, ya que la duración de ejecución del programa es bajo, por lo que cumple con el objetivo de este, además da una respuesta correcta, en donde dada una configuración específica como input, retorna la solución de tal rompecabezas, además, la interfaz gráfica está implementada correctamente, aquí se evidencia el proceso de solución del rompecabezas deslizante. Se puede ver que existe una relación entre la duración de ejecución del programa y qué tan desordenado esté el imput, ambas incrementan, pero no sobrepasan los 0,16 segundos, en el caso de que ese input sí posea solución.

Introducción

Este documento consiste en la documentación técnica de la solución de un problema en Python del proyecto. El cual se enfoca en encontrar una solución programable sobre los rompecabezas deslizantes, el objetivo principal del proyecto es desarrollar una solución que resuelva eficientemente este tipo de rompecabezas y a la vez que retorne la respuesta correcta. Este documento proporciona detalles sobre el enfoque utilizado, el análisis del problema, los algoritmos implementados y algunos ejemplos de corrida del código.

Marco teórico

- **Python 3**
  El lenguaje de programación Python, fue creado por Guido Van Rossum y publicado en 1991 en la versión 0.9.0, este fue inspirado por un lenguaje de computación básico llamado ABC. Python se distingue de los demás programas por ser sencillo y fácil de utilizar, como menciona Robledano (2019):

  > Es un lenguaje de programación versátil multiplataforma y multiparadigma que se destaca por su código legible y limpio. Una de las razones de su éxito es que cuenta con una licencia de código abierto que permite su utilización en cualquier escenario. (párr. 2)[1]

  Este programa, tiene una gran biblioteca de recursos que permiten que el usuario pueda especializar el código en matemáticas, además de ser un lenguaje de gran volumen de datos, favorece extracción y procesamiento del texto, esto lo hace especial porque también es multiplataforma por lo que se puede trabajar desde cualquier sistema operativo, y también, al ser un programa sencillo, con muy pocas líneas de código se puede generar algoritmos complejos. Sin embargo, Python no es muy eficiente para elaborar proyectos como videojuegos, esto es así porque tiene una ejecución de línea a línea lo que lo hace más lento que otros lenguajes de programación. En síntesis, Python es un lenguaje de programación, que permite generar funciones complejas en pocas líneas, pero no es eficiente debido a la gran cantidad de memoria que consume.

- **Bibliotecas utilizadas**
  Pygame:

  Pygame es un conjunto de módulos de Python diseñados específicamente para crear videojuegos y programas multimedia, que se caracteriza por ser gratuito y ser usado para crear juegos de código abierto. Al agregar funcionalidad a la biblioteca SDL, Pygame le permite crear programas multimedia y juegos completos utilizando el lenguaje de programación Python. Como menciona Pygame Community (s.f.):

  > Utiliza código optimizado en C y ensamblador para las funciones principales. El código en C suele ser de 10 a 20 veces más rápido que el código en Python, y el código en ensamblador puede ser fácilmente de 100 veces o más rápido que el código en Python. (párr. 6)[2]

Una de las grandes ventajas de Pygame es su portabilidad, ya que se ejecuta en una amplia variedad de sistemas operativos y plataformas, esto lo convierte en una herramienta muy accesible y versátil para desarrolladores de todo el mundo.
Time:
La biblioteca de python time, funciona para tomar tiempos en los algoritmos, también tiene otras funciones como programar el día, la hora y la fecha de acuerdo al reloj de la computadora, como lo cita a continuación Schmidt (2019) "Una de las funciones principales del módulo time es time(), que devuelve el número de segundos desde el inicio de la «época» como valor de coma flotante" (párr. 4). El principal uso que tiene es medir el tiempo transcurrido en un programa de python, siendo estas mediciones precisas (este puede depender de la plataforma en la que se utilice).

- **IDE´S**
  El IDE utilizado para realizar el código de este proyecto fue Visual Studio Code (también conocido como VS code), este mismo es un editor de código diseñado por Microsoft, es altamente personalizable y extensible, además de que es muy sencillo de usar, lo que lo hace muy popular, ofrece varias herramientas para la edición de código, la depuración, el control de versiones y la integración de extensiones, en resumen es bastante fácil de usar, intuitivo y muy útil.

- **Algoritmo BFS**
  El algoritmo Breadth First Search se utiliza para árboles, y funciona para realizar y marcar un recorrido desde un nodo de origen hacia todos los nodos alcanzables en su nivel más cercano antes de moverse a los niveles más profundos. Como así lo menciona HackerEarth:

  > BFS es un algoritmo de recorrido en el que debe comenzar a recorrer desde un nodo seleccionado (fuente o nodo de inicio) y recorrer el gráfico por capas, explorando así los nodos vecinos (nodos que están directamente conectados al nodo de origen). Luego debe moverse hacia los nodos vecinos del siguiente nivel. (párr. 3) [4]

  Esto quiere decir que se visitará cada vértice en un orden definido, cada vértice se debe visitar solo una vez. También funciona para encontrar el camino más corto entre dos nodos, además, se puede utilizar para verificar la conectividad entre nodos, encontrar componentes conectados en un grafo, determinar si hay ciclos en el grafo y realizar otras tareas que requieran un recorrido completo o búsqueda de amplitud en un grafo.

Descripción de la solución

Para la solución de este proyecto, se utilizaron 5 funciones, pero, primeramente, para la entrada de los números de manera secuencial, primero se hace el input de los números separados por espacios como strings, se convierten a enteros, se crea una matriz nula de 3 filas por 3 columnas y se va llenando la matriz de forma secuencial según como son ingresados los números.

Una de las funciones llamada espacio_vacio, lo que recibe es la "matriz" o mejor dicho el "rompecabezas", que representa la manera en la que están acomodados los números, y lo que hace es recorrer esta matriz y obtener las coordenadas (i, j) del espacio vacío que se representa con un 0.

La siguiente función llamada movimientos_pieza, recibe como parámetro lo mismo que la función anterior, se establece una lista vacía que va a ir almacenando los movimientos o cambios que van realizando las fichas a partir de un rompecabezas dado, seguidamente se llama a la función espacio_vacio para obtener las coordenadas del espacio vacío, luego se crea una lista que se llama movimientos_posibles que lo que contiene son las 4 posibles posiciones (i, j) adyacentes que puede tener el espacio vacío, se itera sobre esta lista de movimientos posibles y se verifica si las coordenadas están dentro de los límites válidos del rompecabezas. Si fuera el caso de que sí están dentro de los límites se crea una copia (nuevo_romp) del rompecabezas actual, esto para no modificar el rompecabezas original con los movimientos que se generan. Luego se intercambia el espacio vacío con el número en la posición (i, j) y se agrega el nuevo_romp a la lista de movimientos. Luego de iterar sobre todas las posiciones en movimientos_posibles, se devuelve la lista movimientos que contiene todas las configuraciones o estados posibles del rompecabezas después de realizar un movimiento o intercambio.

Luego la otra función llamada heu_dist, recibe como parámetro el rompecabezas, se define el rompecabezas objetivo que es el orden al que se quiere llegar realizando movimientos en las piezas, se establece una variable que va a ir almacenando la distancia total estimada entre la configuración del rompecabezas actual y el rompecabezas objetivo, se itera sobre la matriz que representa el rompecabezas y si el número en la posición (i, j) es diferente al número en la posición (i, j) del rompecabezas objetivo, si son diferentes entonces significa que la pieza no está en el lugar donde debería estar y entonces se aumenta la variable que almacena la distancia total en 1, por último se retorna el valor de esta variable, que representa la distancia estimada entre la configuración actual y el estado objetivo. Cuanto menor sea esta distancia, más cercano estará el rompecabezas del estado objetivo, y se utilizará para priorizar los movimientos que acerquen al rompecabezas a ese estado.

La siguiente función es la función BFS que lleva la cola y que utiliza la búsqueda por anchura para encontrar la solución correcta al rompecabezas. Primero se establece una lista, donde se van a almacenar las configuraciones del rompecabezas que ya se han hecho o "visitado", luego se crea la cola con un elemento inicial, esta cola va a ir almacenando tripletes que son la distancia de la heurística, la configuración actual del rompecabezas y el "recorrido" que se ha hecho hasta ese momento, este recorrido son todos los rompecabezas que muestran como se han ido moviendo las piezas. Luego mientras la cola no esté vacía, se ordena la cola en función de la distancia estimada usando la función sort(), esto para que los rompecabezas con menos distancia estimada estén al inicio de la cola, seguidamente se extrae el primer elemento de la cola que va a contener la distancia heurística, la configuración actual del rompecabezas y el recorrido hasta ese momento, se convierte la configuración del rompecabezas actual en una tupla para poder verificar si esta configuración ya ha sido visitada, si ya fue visitada entonces se salta a la siguiente iteración y se ignora esta configuración, pero si la configuración no ha sido visitada, entonces se agrega a la lista de visitados para evitar enciclamientos, seguidamente se verifica si la configuración actual es igual al rompecabezas objetivo, si sí es igual, entonces ya se habría encontrado la solución y se retorna el recorrido hasta esa configuración. Si no es igual, entonces se generan más movimientos a partir de la configuración

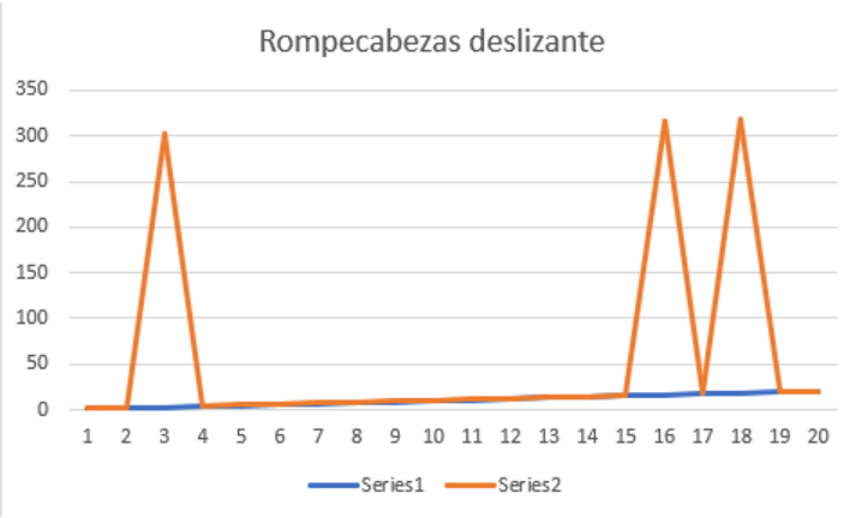
actual del rompecabezas, y se itera sobre cada movimiento en estos movimientos generados, por cada movimiento se agrega a la cola una tupla que contiene la distancia heurística estimada del movimiento, la nueva configuración del rompecabezas después del movimiento y el recorrido actualizado que incluye el movimiento realizado. Si al final se vacía la cola y no se encuentra ninguna solución se retorna una lista con un 0 para indicar que no hay solución válida.

La función de la interfaz se encarga de dibujar la representación gráfica del rompecabezas en una ventana usando pygame, primeramente, se llena la ventana con color blanco, esto para limpiar la ventana antes de dibujar el rompecabezas, luego se calculan las dimensiones de cada cuadro en el rompecabezas dividiendo las dimensiones de la ventana entre 3, esto para que cada cuadro tenga el mismo tamaño en función del tamaño de la ventana. Se itera sobre la longitud en filas y columnas de la matriz y se obtiene el número que está en la posición (i, j) en el rompecabezas y se verifica si el número no es 0, si no es 0 entonces significa que hay una pieza en esa posición y entonces se debe dibujar un cuadro con el número que representa esa ficha en el centro, si el número es 0, significa que esa posición representa el espacio vacío. En este caso, se dibuja un cuadro de un color diferente para representar el espacio vacío.

Y por último se establece que la solución va a ser una llamada a la función BFS que lleva la cola, si ya se hizo el input de los números que representan el rompecabezas y si hay solución para esa entrada, entonces se llama a la función para dibujar el rompecabezas, luego se itera sobre la solución y se va dibujando cada configuración de rompecabezas que se obtuvo para llegar al objetivo.

Resultados de las pruebas

Se realizaron diversos casos de prueba en el código realizado, y a estos se les tomó el tiempo de duración para dar una solución válida, y con esto realizar la gráfica de los casos de prueba. Como se muestra en la siguiente figura:



En la ejecución de algunos casos, se eleva el tiempo y en otros se mantiene constante dependiendo de la configuración dada del rompecabezas, en total se tomaron diversas

configuraciones, algunas en las cuales no existía una solución posible y otras en las que evidentemente sí existe, si fuera el caso de que no haya solución, el tiempo de ejecución del programa se eleva, a continuación se aprecian los casos tomados y los tiempos de culminación de los mismos:

132807456	0,14
102837456	0,12
152837406	300
052137486	0,11
210537486	0,13
215037486	0,11
215437086	0,12
817263540	0,11
132087456	0,11
876154320	0,12
087654321	0,1
051273486	0,12
012345678	0,15
876543210	0,16
610453782	0,15
642175830	300
537206184	0,15
438672015	300
362815407	0,15
837156240	0,13

Estos casos se probaron en la terminal de una computadora marca MAC y sobre el sistema operativo IOS.

Conclusiones

Como análisis general de los resultados obtenidos, se tiene que este algoritmo se ejecuta de manera eficiente con cualquier configuración inicial de rompecabezas dado, esto se debe a que se está ordenando la cola usando la función sort(), para que los rompecabezas con menos distancia estimada al rompecabezas objetivo estén al inicio de la cola y por lo tanto que se comparen primero.

En lo que respecta a la eficiencia del proyecto, este podría ser más eficiente implementando funciones como lambda o set, con esto se logra disminuir el tiempo de ejecución del algoritmo de manera significativa.

Aprendizajes

- **Josué Quintana**
  Durante el proyecto, se aprendió mucho sobre cómo abordar problemas y trabajar en equipo para encontrar soluciones efectivas. Además, antes de comenzar el proyecto, se invirtió tiempo y esfuerzo en comprender claramente el problema que se estaba tratando de resolver para así intentar encontrar una solución más fácil y efectiva. El equipo

aprendió a comunicarse claramente, escuchar activamente y proporcionar retroalimentación constructiva para garantizar que todos los miembros del equipo aportaran ideas válidas para la resolución del proyecto.

- **Angie Esquivel**
  Uno de los aprendizajes más importantes fue el valor de la responsabilidad compartida. Aprendimos que al asumir responsabilidades individuales y trabajar juntos hacia la misma meta, logramos una mayor eficiencia y calidad en nuestros resultados. Además, esta responsabilidad compartida nos permitió construir confianza mutua y fortalecer nuestra comunicación como equipo.

- **Ana Murillo**
  En este proyecto, aprendí a escuchar a mis compañeros y a tener paciencia, también a ser comprensiva con la forma en la que trabajan otros compañeros. Entendí que este trabajo no se hubiese podido lograr sin alguno de los tres, porque a pesar de que a los tres nos cuesta un poco, nos supimos apoyar en los otros para que este trabajo diera un buen resultado, y cada uno aportó de forma diferente en este proyecto. Pero no solo aprendí eso, sino también logré entender algunos de los contenidos del curso, y cómo trabajar con estos para la realización del proyecto.

Referencias bibliográficas

[1]A, Robledano. (2019). Qué es Python: características, evolución y futuro.OpenWebinars. https://openwebinars.net/blog/que-es-python/

[2]Pygame Community. (s.f.). About - wiki. pygame wiki. https://www.pygame.org/wiki/about

[3]Schmidt, E.(2019). time - Hora del reloj. Ernesto Rico Schmidt. https://rico-schmidt.name/pymotw-3/time/index.html

[4] HackerEarth. (s. f.). Breadth First Search (BFS). HackersEarth. https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/