Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2022.

ROBOT PISOS ARTESANALES

202003585 – Diego Andrés Huite Alvarez

Resumen

El proyecto *Robot pisos artesanales* es un software elaborado para la empresa *pisos artesanales S.A.* Esta empresa se encarga de colocar pisos alrededor de todo el país. Estos pisos pueden ser colocados con diversos patrones de azulejos de colores blanco y negro. El cliente, puede decidir si quiere cambiar su patrón actual por uno nuevo. Para llevar a cabo este cambio de patrones se desarrolló un robot optimizado para calcular el coste mínimo para pasar de un patrón a otro, así como también mostrar los pasos necesarios para pasar de un patrón a otro. Para resolver este problema se hizo uso de *listas enlazadas simples* y listas doblemente enlazadas, dos *estructuras de datos* fundamentales. Se hizo uso del lenguaje de programación orientado a objetos conocido como *Python* y una herramienta de graficación conocida como *graphviz.* La graficación consiste en un archivo con extensión *.dot* y se muestra en forma de imagen *.png.*

Palabras clave

Estructura de datos: Forma de organizar datos en un computador para usarlos de manera eficiente

Lista enlazada simple: Estructura de datos que consiste en una secuencia de nodos y punteros.

Python: Lenguaje POO creado en 1991.

Abstract

The *Robot handmade floors* project is a software developed for the company handmade floors S.A. This company is responsible for placing floors around the country. These floors can be laid with various patterns of black and white colored tiles. The client can decide if he wants to change his current pattern for a new one. To carry out this change of patterns, an optimized robot was developed to calculate the minimum cost to go from one pattern to another, as well as to show the steps necessary to go from one pattern to another. To solve this problem, use was made of *single linked lists* and doubly linked lists, two fundamental *data structures*. The object-oriented programming language known as *Python* and a graphing tool known as *graphviz* were used. The graph consists of a file with a .dot extension and is shown as a .png image.

Keywords

Data Structure: A way of organizing data in a computer so that it can be used efficiently.

Simple Linked List: A data structure consisting of a sequence of nodes and pointers.

Python: OOP language created in 1991.

Introducción

Desde siempre se ha buscado reducir los costos en cualquier ámbito, desde, compra de materiales, compra de productos básicos, etc. Buscando obtener el máximo provecho del dinero. En la actualidad vivir se ha vuelto más caro.

Es por eso por lo que, se ha desarrollado un software capaz de minimizar costos en la remodelación de pisos. Se busca que el cliente pueda transformar un patrón de piso a otro. Este software es capaz de indicarle al usuario que operaciones deben de realizar para transformar un patrón a otro con el coste mínimo de quetzales. Los azulejos pueden ser volteados, ya que al reverso poseen el color opuesto y también pueden ser intercambiados, pero solamente con los azulejos adyacentes, exceptuando los intercambios en diagonal. Cada una de estas operaciones posee un costo. Para tener una mejor visión, el usuario puede ver como lucen los patrones por medio de una imagen.

1. Identificación del problema y solución aplicada

1.1 El problema

Al momento de instalar un piso, este puede ser de tamaño MxN y el patrón puede ser cualquiera. Cuando el cliente desee transformar el patrón que desee se debe obtener la solución paso a paso para pasar un patrón origen a un patrón destino.

Voltear un piso puede costar F quetzales mientras que intercambiar uno puede costar S quetzales. F puede ser mayor a S o igual y viceversa.

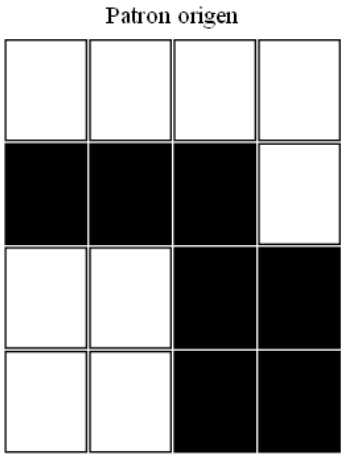

Figura 1. Ejemplo de un piso con un patrón origen.

Fuente: elaboración propia.

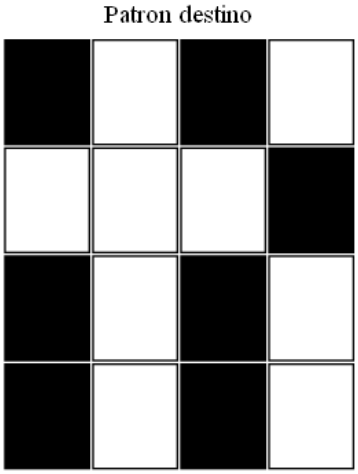

Figura 2. Ejemplo de un piso con un patrón destino.

Fuente: elaboración propia.

Pero ¿Cómo construimos un algoritmo sabiendo que no hará movimientos innecesarios y obtener el coste mínimo?

1.2 Análisis para solucionar el problema

Para solucionar este problema se hicieron muchísimas pruebas para encontrar el algoritmo que lo soluciona. Uno podría pensar que se debe empezar con la operación más barata, y usar solamente esta operación, pero esto no siempre resultará.

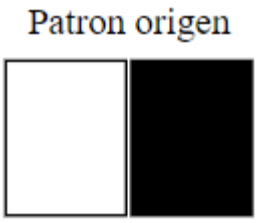

Figura 3. Ejemplo de un piso con un patrón origen.

Fuente: elaboración propia.

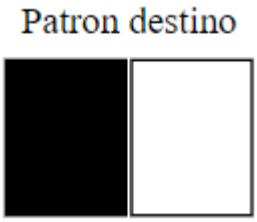

Figura 4. Ejemplo de un piso con un patrón destino.

Fuente: elaboración propia

Considere los pisos de la figura 3 y 4.

Sea S el costo para voltear un azulejo.

Sea F el costo para intercambiar un azulejo.

Comenzaremos tomando en cuenta la cantidad de azulejos de negros y blancos de cada patrón, así podríamos determinar si es necesario voltear alguna celda. En este ejemplo tenemos la misma cantidad de azulejos blancos y negros en ambos patrones, por lo que tenemos dos opciones.

1. Resolver este problema usando solamente intercambios.

2. Resolver este problema usando solamente volteando las celdas erróneas.

Cuando la cantidad de celdas negras y blancas son distintas también tenemos dos opciones:

1. Resolver el problema usando intercambios y volteos de celda.

2. Resolverlo solamente volteando celdas.

Debemos considerar también como serán los costes de la operación. Se pueden dar tres casos distintos:

Caso 1:

$$S = F$$

Para este caso definiremos el valor de S y de F como 1 quetzal.

Si resolvemos este caso intercambiando las celdas obtenemos un costo a pagar de 1 quetzal. Pero si decidimos resolver el problema volteando las celdas obtenemos un costo de 2 quetzales. Por lo que se determina que intercambiar es nuestra mejor opción en esta situación. Cuando los costes de las operaciones sean iguales, debemos de priorizar el número mínimo de movimientos para resolver este caso.

Caso 2:

$$S < F$$

Definimos S como 1 y F como 3

En este caso vemos que si resolvemos el problema solo con intercambios el coste total es de 1, mientras que si lo hacemos con volteos de celda se deben pagar 6 quetzales. Nos conviene intercambiar celdas.

Caso 3:

$$F < S$$

Definimos F como 2 y S como 3.

Al solucionar solo con intercambios obtenemos que se deben pagar 3 quetzales, mientras que solo volteando azulejos debemos de pagar 4 quetzales, por lo que aquí se demuestra que no siempre es mejor resolver el problema usando solamente la opción más barata.

Estos tres casos también se presentan cuando el número de azulejos blancos y negro no son los mismos. Con la diferencia de que sabemos cuántas celdas debemos voltear.

1.3 Algoritmo solución

La solución implementada se define por el siguiente algoritmo:

1. Buscar el primer azulejo erróneo en el patrón origen.
2. Buscar el segundo azulejo erróneo en el patrón origen del color distinto del primer azulejo erróneo. Este debe de ser el más cercano al azulejo origen.
3. Construir un "camino" entre estos azulejos.
4. Si estos azulejos son adyacentes, debemos de intercambiarlos de una vez. Y repetir el paso 1
5. Determinar si el camino entre estos azulejos es de un mismo color o es de colores combinados.
6. Si el camino es de un mismo color, debemos trasladar el azulejo del color opuesto al color del camino hacia el azulejo origen o destino. Repetir
7. Si el camino es de colores combinados debemos buscar el primer azulejo del color distinto al del azulejo origen, y reducirlo al camino de un mismo color. Repetimos
8. Repetir desde el paso 1.
9. Si se llega al caso en el que ya no hay una segunda celda errónea significa que debemos de girar estas.

Como fue mencionado en la sección 1.2, también se puede solucionar solamente girando celdas, por lo que se combinaran intercambios

Este algoritmo surgió luego de incontables pruebas para hallarlo.

1.4 Ejemplo usando el algoritmo solución.

Consideremos los pisos de las figuras 1 y 2. Al aplicar los pasos 1, 2 y 3 obtenemos lo siguiente:

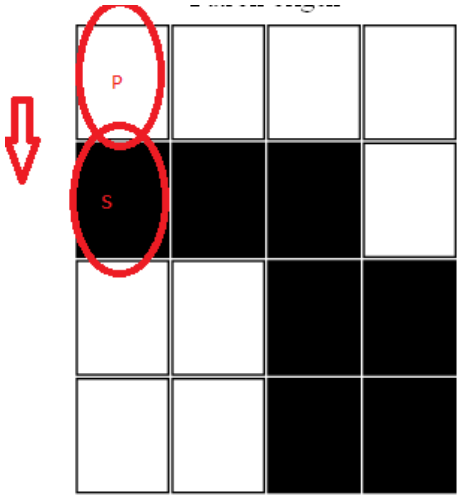

Figura 5. Aplicando pasos 1, 2 y 3 a un piso

Fuente: elaboración propia.

El primer azulejo erróneo está denotado por la letra P, mientras que el segundo con la letra S. Al ir al paso 4, hacemos el intercambio.

Volvemos a analizar y se repite el mismo caso anterior.

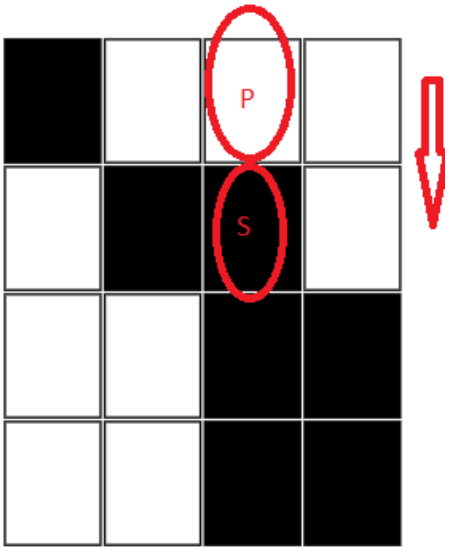

Figura 6. Aplicando pasos 1, 2 y 3 a un piso

Fuente: elaboración propia.

Al hacer el intercambio. Repetimos el algoritmo y obtenemos lo siguiente:

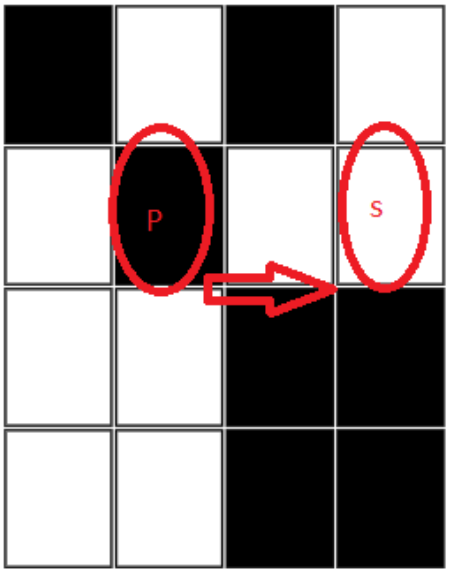

Figura 7. Aplicando pasos 1, 2 y 3 a un piso

Fuente: elaboración propia.

Continuamos con el paso 4 y vemos que el camino entre estos dos es solamente de color blanco. Siguiendo el paso 5 movemos el azulejo P

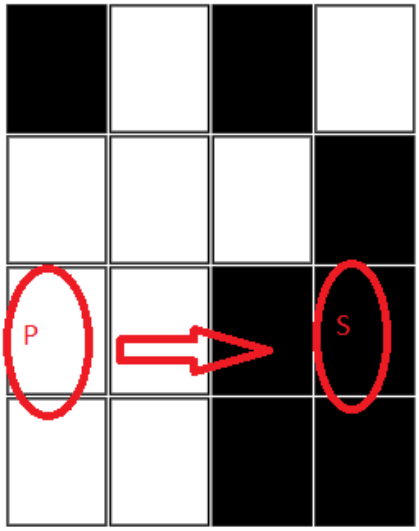

Figura 8. Aplicando pasos 1, 2 y 3 a un piso

Fuente: elaboración propia.

Repetimos el algoritmo y llegamos al caso donde hay colores combinados entre el azulejo origen y destino.

En este caso debemos aplicar el paso 7 y simplificando el ca mino quedaría de la siguiente manera:

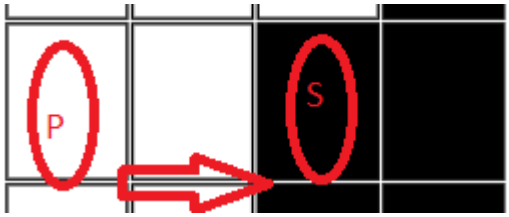

Figura 9. Simplificando camino entre azulejos.

Fuente: elaboración propia.

Se ha reducido el caso hasta el paso 6 y continuamos así hasta resolver todo el patrón.

2. Estructuración del software

2.1. Clases utilizadas para estructurar el software

1. Celda
2. Patrón
3. Piso
4. NodoCelda
5. NodoPatron
6. NodoPiso
7. Tablero_lineal
8. Tablero
9. ListaPatron
10. ListaPiso
11. NodoFila

Los nodos de celdas y de filas son nodos doblemente enlazados, esto es así para que sea más fácil la navegación entre azulejos y filas de un piso. Mientras que los nodos de pisos y patrones son simples, ya que solamente necesitamos almacenar la información de estos, mas no realizar operaciones.

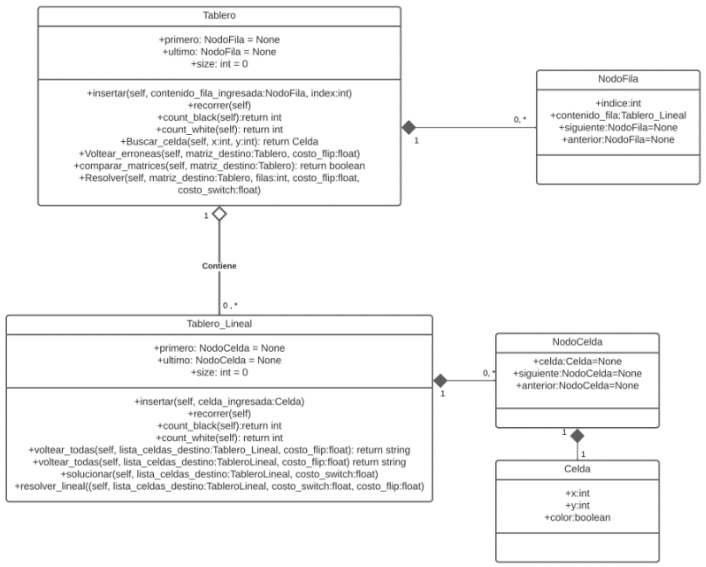

Figura 10. Diagrama de clases de los tableros y Nodos Celda

Fuente: elaboración propia.

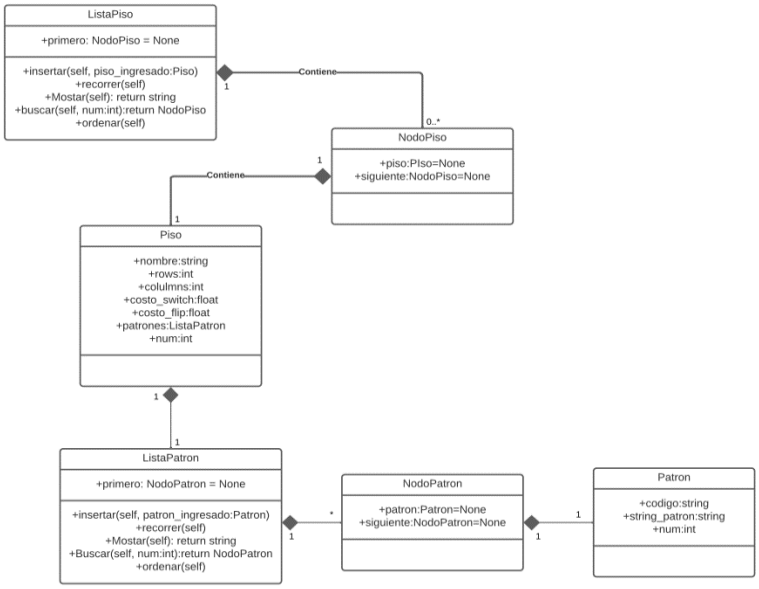

Figura 11. Diagrama de clases de las listas de pisos, patrones, etc.

Fuente: elaboración propia

Conclusiones

Las estructuras de datos simplifican mucho el trabajo al manejar mucha información. Son de gran ayuda para tener un control de una cantidad indeterminada de datos, es una gran ventaja contra los arreglos o arrays. Ya que estos solo permiten almacenar solamente un tipo de dato, al contrario, las listas enlazadas permiten almacenar distintos tipos de datos en un nodo. Sin embargo, también cuentan con una gran desventaja frente a los arreglos, y es el uso de memoria. Las listas enlazadas consumen más memoria que un arreglo. Esto podría causar conflicto en un hardware con muy poca memoria.

Referencias bibliográficas

1. Estructura de datos. (4 de marzo de 2022). En Wikipedia. https://es.wikipedia.org/w/index.php?title=Estructura_de_datos&oldid=141752350
2. Lista enlazada. (4 de marzo de 2022). En Wikipedia. https://es.wikipedia.org/w/index.php?title=Lista_enlazada&oldid=141959533
3. Python. (4 de marzo de 2022). En Wikipedia. https://es.wikipedia.org/w/index.php?title=Python&oldid=141916147