
Pisos Artesanales, S.A.

202001568 1 – Nathan Antonio Valdez Valdez

Resumen

La empresa “Pisos Artesanales, S.A.” es una empresa de venta de pisos cerámicos que estos poseen distintos patrones. Cada piso tiene diferentes patrones con diseños distintos por lo que el usuario que desea comprar algún piso puede elegir el patrón que desean y para ello necesitan un robot que automatice el trabajo de seleccionar el patrón que el usuario indique y cambiarlo, se le estará cobrando al usuario por cada modificación de patrón que se le haga. Por ello se creó este código de lectura de archivos, implementación de listas enlazadas, doblemente enlazadas, graficar los patrones entrantes y salientes, instrucciones detalladas de los pasos al realizar el cambio de patrón; Este código implementa un algoritmo para que el robot analice los pasos a convertir el cambio de patrón y optimizar los gastos finales para el usuario para obtenga el coste mínimo posible a la hora de comprar su piso con el patrón a su gusto.

Palabras clave

Listas Enlazadas, Listas Doblemente Enlazadas,
Graphviz, Archivo XML, TDA, Nodos

Abstract

The company "Pisos Artesanales, S.A." is a company that sells ceramic floors that have different patterns. Each floor has different patterns with different designs, so the user who wants to buy a floor can choose the pattern they want and for this they need a robot that automates the work of selecting the pattern that the user indicates and changing it, the user will be charged user for each pattern modification made to it. For this reason, this file reading code was created, implementation of linked lists, doubly linked, graphing the incoming and outgoing patterns, detailed instructions of the steps when changing the pattern; This code implements an algorithm so that the robot analyzes the steps to convert the pattern change and optimize the final expenses for the user to obtain the minimum possible cost when buying their flat with the pattern they like.

Keywords

*Linked Lists, Doubly Linked Lists,
Graphviz, XML File, TDA, Nodes*

Introducción

Crear un software el cual se debe de utilizar mediante línea de comandos y fácil de utilizar. El programa debe ser capaz de cargar un archivo XML y procesar su información para almacenarla en listas enlazadas.

Dicha aplicación recibe un archivo XML que contiene información sobre los pisos en existencia de la empresa que solicito el algoritmo. El usuario puede elegir la ruta donde se encuentra el archivo XML para cargarlo, el usuario podrá elegir el piso que desea comprar y se le indican los patrones que el mismo tiene, si el usuario quisiera cambiar el patrón que viene por defecto en el piso seleccionado se le indicara al programa que realice el cambio mostrándole gráficamente los resultados en una imagen PNG

Desarrollo del tema

El problema que se nos presenta es el siguiente: “La empresa ha comprado un robot especializado capaz de colocar los pisos de dimensiones $R \times C$ con cualquier patrón, combinando azulejos del lado blanco con azulejos del lado negro. Se le ha solicitado realizar un programa que garantice que, al modificar un patrón en un piso existente, el costo de hacer esta modificación sea el mínimo posible para optimizar el uso del robot especializado adquirido para este fin.”

Para la solución de este problema se usó el lenguaje Python, el cual implementamos la librería “Element Tree” para el manejo de la información que se ingresa al programa por medio de un archivo XML. A continuación, se explicará detalladamente todas las funciones que el sistema posee, así como también el algoritmo aplicado para realizar todo el proceso.

A. Funciones del sistema

1. Carga de archivo XML
2. Mostrar datos cargados
3. Seleccionar un piso en especifico
4. Graficar el patrón del piso que viene por defecto
5. Cambiar el patrón por un nuevo patrón que el usuario solicite
6. Graficar el patrón solicitado por el usuario
7. Mostrar instrucciones paso a paso de como se cambió el patrón.
8. Salir del sistema

B. Explicación del Programa

Al iniciar el programa lo que se le mostrará al usuario es el menú principal de opciones que tenemos, el usuario deberá ingresar con el teclado numérico la opción de que desee ejecutar. Si el usuario ingresa una opción que no está dentro del menú el programa mostrara nuevamente el menú hasta que el usuario digite una opción valida

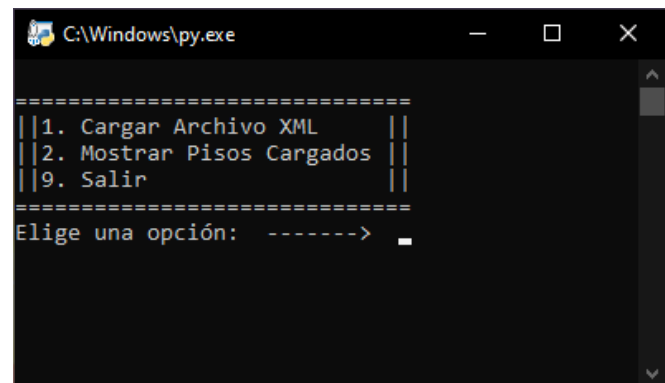


Figura 1. Menú Principal

Fuente: elaboración propia

1. Cargar Archivo XML

Al ejecutar esta función de nuestro menú principal se abrirá automáticamente el explorador de archivos de su sistema operativo para que el usuario pueda seleccionar cómodamente el archivo que desea cargar al programa, por defecto está configurado para que solo nos muestre archivos estrictamente con extensión “.xml” y carpetas, pero se puede cambiar la opción a mostrar todos los archivos de su almacenamiento y se mostraran todos los archivos con cualquier tipo de extensión

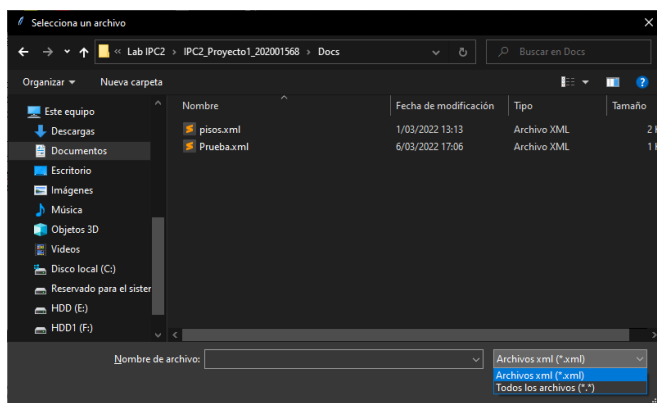


Figura 2. FileChooser

Fuente: elaboración propia

El programa automáticamente leerá el archivo seleccionado y lo almacenara en la memoria del programa por medio de una lista enlazada que contiene dentro de ella Nodos que en estos se almacena la información de cada piso ingresado que tiene como atributos:

- Nombre
- Numero de Filas
- Numero de Columnas
- Costo de Volteo
- Costo de deslizamiento
- Lista de Patrones

Los patrones que tiene cada piso se almacenan dentro de una lista especial para ellos que esta como un atributo más del nodo piso. Dentro de esta lista específica para los patrones, cada uno de los patrones tiene sus propios atributos:

- Código del Patrón
- Patrón en una cadena

Cada patrón se almacena con su Código que funciona como un ID y el patrón que fue filtrado a la hora de leer el archivo xml borrándole los espacios y saltos de líneas

Todo esto se almacena en listas enlazadas que sus apuntadores fueron asignados conforme a los pisos que están ingresando

Si en dado caso el archivo xml tiene errores en los datos de un piso, el programa mostrara el mensaje de que piso tiene el error y no lo cargara a la memoria de nuestro programa, por ejemplo “No se cargaron los datos correctamente del piso {nombre del piso}” y se deberá corregir el archivo xml.

2. Mostrar pisos cargados

Al ingresar a esta opción de nuestro menú principal se nos mostraran todos los pisos cargados con su información detalladamente.

Luego de mostrar todos los pisos cargados al programa el usuario podrá elegir el piso que desee y dentro de el podrá hacer las operaciones de intercambiar o mostrar su patrón por defecto

```

C:\Windows\py.exe
Nombre: ejemplo01 R- 2 C- 4 F- 1 S- 1
con los patrones:
codigo= cod11 patron- WBWBWBBW
codigo= cod12 patron- BWBWWWW
-----
Nombre: ejemplo02 R- 3 C- 3 F- 1 S- 1
con los patrones:
codigo= cod21 patron- WBBWBWW
codigo= cod22 patron- WWWBWWB
-----
Nombre: ejemplo03 R- 1 C- 5 F- 1000 S- 1
con los patrones:
codigo= cod31 patron- WBBBB
codigo= cod32 patron- BBBW
-----
Nombre: atol R- 3 C- 3 F- 1000 S- 1
con los patrones:
codigo= big patron- BWWBWWB
codigo= joo patron- WWBWWBBW
-----
||-> Ingrese el nombre del piso para seleccionar ||
||9. Regresar al Menu Principal ||
=====
Elige una opción: ----->

```

Figura 3. Archivos Cargados

Fuente: elaboración propia

La información que contiene cada piso es:

- Nombre = Nombre del piso
- R = Numero de filas
- C = Numero de columnas
- F = Costo de invertir color
- S = Costo de deslizar color
- Patrones asignados a dicho piso

El usuario desde este punto podrá regresar al menú principal o seleccionar el piso que desee, El usuario deberá ingresar el nombre del piso correctamente como se muestra, de caso contrario el programa mostrara el mensaje “No se encontró el piso {nombre ingresado}” y volverá a desplegar el mismo menú en el que se encuentra

Una vez seleccionado el piso correctamente se volverá a mostrar el nombre del piso y los patrones que tiene consigo.

Si el usuario desea ver el patrón por defecto del piso que ha seleccionado puede elegir la opción 1 y el programa abrirá automáticamente la imagen del patron con el visualizador predeterminado de su sistema operativo.

```

Seleccinar C:\Windows\py.exe
piso atol seleccionado
con los siguientes patrones:
codigo= big patron- BWWBWWB
codigo= joo patron- WWBWWBBW
=====
||1. Mostrar Graficamente el Piso ||
||-> Para cambiar el patron por favor ingrese el codigo ||
|| del patron que desea cambiar ||
||9. Atras ||
=====
Elige una opción: ----->

```

Figura 4. Piso Seleccionado

Fuente: elaboración propia

Si el usuario deseara cambiar el patrón por defecto de un piso primero deberá ingresar el código del patrón que se cambiara, si el usuario ingresa un código incorrecto el programa mostrara el mensaje 'No se encontró el patrón {patron_ingresado}'

Al seleccionar correctamente el código del patrón inicial se desplegará otro mensaje con el código del patrón y su patrón. Luego de esto el usuario deberá ingresar el código del segundo patrón que el programa seguirá de ejemplo para cambiar

```

C:\Windows\py.exe
codigo big seleccionado
Patron: BWWBWWB
=====
||1. Mostrar Graficamente el Piso ||
||->.Para cambiar de patron seleccione ||
|| otro codigo de patron con el cual intercambiar ||
||3. Mostrar Instrucciones ||
||9. Atras ||
=====
Elige una opción: ----->

```

Figura 5. Patron Inicial Seleccionado

Fuente: elaboración propia

Al seleccionar correctamente el código del segundo patrón empieza el algoritmo que se diseñó para resolver el problema.

Lo primero que hace nuestro algoritmo es verificar al mismo tiempo los 2 patrones y si las celdas contienen el mismo color este las bloquea para que no las pueda mover con los siguientes movimientos que va a realizar.

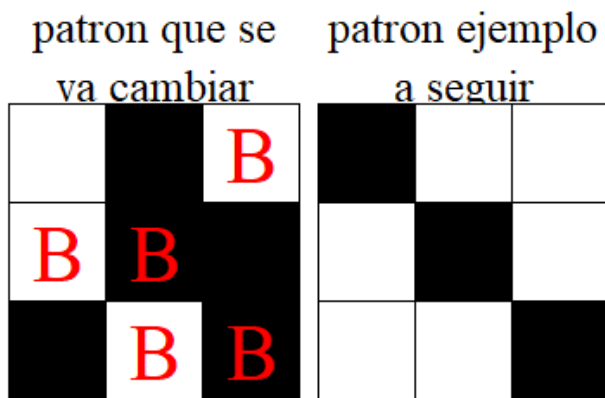


Figura 6. Bloqueo de celdas

Fuente: elaboración propia

```

88
89 def RefreshPatt(L1:Cell_List, L2:Cell_List):
90     tmp1 = L1.First
91     tmp2 = L2.First
92     while tmp1 is not None:
93         if tmp1.getColor() == tmp2.getColor():
94             tmp1.BlockCell()
95             tmp1 = tmp1.getNext()
96             tmp2 = tmp2.getNext()
97

```

Figura 7. Función Bloquear Celdas

Fuente: elaboración propia

El siguiente paso para nuestro algoritmo es verificar los costes de deslizar celdas e invertir celdas para que a la hora de hacer los movimientos este hago por defecto la operación más barata para que el usuario pague el coste mínimo.

Para realizar el movimiento de deslizar una celda se crearon dos funciones diferentes, una función esta hecha específicamente para que el algoritmo deslice las celdas de izquierda a derecha (SlidePattt) y la otra función para que deslice de arriba hacia abajo (SlideDown)

SlidePatt

Esta función recorre las dos listas de patrones que tenemos y va comparando cada una de las celdas con su respectiva celda “ejemplo” que tenemos dentro de la segunda lista, para realizar el movimiento tendremos que hacer unas validaciones para que este se cumpla:

- Si el color es diferente
- Si el color de la siguiente celda es diferente
- Si la celda actual está bloqueada
- Si la celda siguiente está bloqueada
- Si la posición en Y de la celda actual es igual a la posición en Y de la celda siguiente

Si se cumplen todas estas validaciones el programa procede a realizar el cambio de celdas(deslizar), para deslizar la celda lo primero que se hace es guardar el color de la celda actual en una variable temporal, lo siguiente seria modificar el color de la celda actual con el color de la celda siguiente, modificar el color de la celda siguiente con el color anterior que tenía la celda actual que lo guardamos en nuestra variable temporal, por último se bloquea la celda actual ya que contiene el color “ejemplar” que proviene de nuestra segunda lista

SlideDown

En esta función lo que cambia es el recorrido de la primera celda ya que nuestra celda para deslizar se tiene que encontrar justo debajo de la celda actual para ello se agregaron más validaciones para que se pueda cumplir el deslizamiento hacia abajo.

- Si la celda actual está bloqueada
- Si la celda siguiente tiene la misma posición en X que la celda actual
- Si la celda siguiente tiene la posición de la celda actual + 1
- Si la celda siguiente está bloqueada
- Si el color de la celda actual es diferente a la celda siguiente

Si todas estas validaciones se cumplen se realiza el mismo proceso de intercambio que la función anterior

Invert Cell

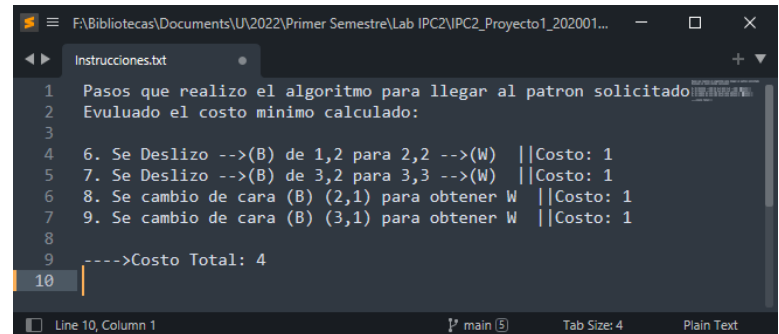
Esta función es mas sencilla que las anteriores y lo podemos ver por las validaciones que contiene:

- Si celda actual esta bloqueada
- Si celda actual contiene diferente color a la celda “ejemplar” de la segunda lista

Si estas dos validaciones se cumplen se procede a modificar el color de la celda actual por el color de la celda “ejemplar” y se bloquea la celda actual

Al realizar el cambio de patrón el programa mostrara el mensaje “Intercambio exitoso” y si el usuario desea ver los pasos que el sistema realizo para hacer el intercambio y el costo total a pagar por el cambio de patrón deberá seleccionar la opción 3 que se

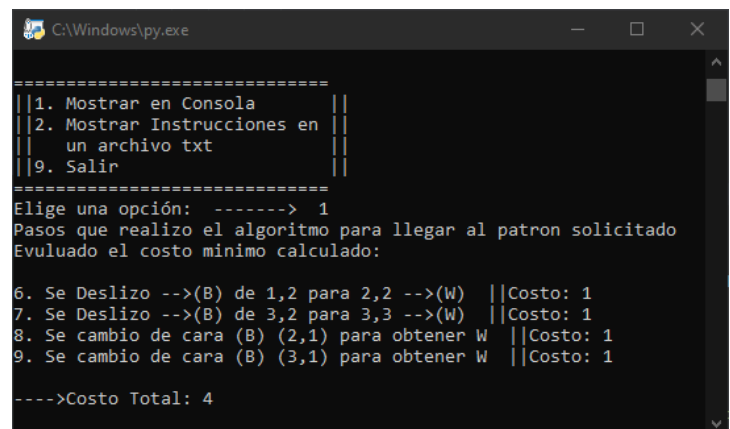
encuentra en ese menú, al seleccionar esta opción el usuario podrá escoger si desea ver las instrucciones en consola o si desea extraerlas a un archivo txt que se generara y abrirá automáticamente con su visualizador de texto predeterminado.



```
1 Pasos que realizo el algoritmo para llegar al patron solicitado
2 Evaluado el costo minimo calculado:
3
4 6. Se Deslizo -->(B) de 1,2 para 2,2 -->(W) ||Costo: 1
5 7. Se Deslizo -->(B) de 3,2 para 3,3 -->(W) ||Costo: 1
6 8. Se cambio de cara (B) (2,1) para obtener W ||Costo: 1
7 9. Se cambio de cara (B) (3,1) para obtener W ||Costo: 1
8
9 ---->Costo Total: 4
10
```

Figura 8. Instrucciones en consola

Fuente: elaboración propia



```
C:\Windows\py.exe
=====
||1. Mostrar en Consola ||
||2. Mostrar Instrucciones en ||
|| un archivo txt ||
||9. Salir ||
=====
Elige una opción: -----> 1
Pasos que realizo el algoritmo para llegar al patron solicitado
Evaluado el costo minimo calculado:

6. Se Deslizo -->(B) de 1,2 para 2,2 -->(W) ||Costo: 1
7. Se Deslizo -->(B) de 3,2 para 3,3 -->(W) ||Costo: 1
8. Se cambio de cara (B) (2,1) para obtener W ||Costo: 1
9. Se cambio de cara (B) (3,1) para obtener W ||Costo: 1

---->Costo Total: 4
```

Figura 9. Instrucciones en archivo txt

Fuente: elaboración propia

```
SlidePatt(L1: Cell_List, L2: Cell_List, Floor: Floor):
    tmp1 = L1.First
    tmp2 = L2.First
    tmp1Next: Cell = tmp1.getNext()

    while tmp1Next is not None:
        if tmp1.getColor() != tmp2.getColor() and tmp1.getColor() != tmp1Next.getColor() :
            if tmp1.getBlock() == False and tmp1.getPosY() == tmp1Next.getPosY() and tmp1Next.getBlock() == False:
                global Instructions
                global MovCount
                global SlideCount
                SlideCount += 1
                MovCount += 1
                Instructions += '{}. Se Deslizo -->({}) de {},{} para {},{} -->({}) ||Costo: {}\\n'.format(
                    MovCount, tmp1.getColor(), tmp1.getPosX(), tmp1.getPosY(), tmp1Next.getPosX(),
                    tmp1Next.getPosY(), tmp1Next.getColor(), Floor.getSlideCost())

                tmp = tmp1.getColor()
                tmp1.setColor(tmp1Next.getColor())
                tmp1Next.setColor(tmp)
                tmp1.BlockCell()
                RefreshPatt(L1, L2)

            tmp1 = tmp1.getNext()
            tmp2 = tmp2.getNext()
            tmp1Next: Cell = tmp1.getNext()
```

Figura 10. Código SlidePatt

Fuente: elaboración propia

```
SlideDown(L1: Cell_List, L2: Cell_List, Floor: Floor):
    tmp1 = L1.First
    tmp2 = L2.First

    while tmp1 is not None:
        if tmp1.getBlock() == False:
            tmp1Down: Cell = tmp1.getNext()
            while tmp1Down is not None:
                if tmp1Down.getPosX() == tmp1.getPosX() and tmp1Down.getPosY() == tmp1.getPosY()+1:
                    if tmp1.getBlock() == False and tmp1.getColor() != tmp1Down.getColor() and tmp1Down.getBlock() == False:
                        global Instructions
                        global MovCount
                        global SlideCount
                        SlideCount += 1
                        MovCount += 1
                        Instructions += '{}. Se Deslizo -->({}) de {},{} para {},{} -->({}) ||Costo: {}\\n'.format(
                            MovCount, tmp1.getColor(), tmp1.getPosX(), tmp1.getPosY(), tmp1Down.getPosX(), tmp1Down.getPosY(), tmp1Down.getColor(), Floor.getSlideCost())

                        tmp = tmp1.getColor()
                        tmp1.setColor(tmp1Down.getColor())
                        tmp1Down.setColor(tmp)
                        tmp1.BlockCell()
                        RefreshPatt(L1, L2)

                    tmp1Down = tmp1Down.getNext()

            tmp1 = tmp1.getNext()
            tmp2 = tmp2.getNext()
```

Figura 11. Código SlideDown

Fuente: elaboración propia


```
InvertCell(L1: Cell_List, L2: Cell_List, Floor: Floor):  
tmp1 = L1.First  
tmp2 = L2.First  
while tmp1 is not None:  
    if tmp1.getBlock() == False and tmp1.getColor() != tmp2.getColor():  
        global Instructions  
        global MovCount  
        global FlipCount  
        FlipCount += 1  
        MovCount += 1  
        Instructions += '{}. Se cambio de cara ({{}) ({{},{{}) para obtener {{} ||Costo: {{}\n'.format(  
            MovCount, tmp1.getColor(), tmp1.getPosX(), tmp1.getPosY(), tmp2.getColor(), Floor.getFlipCost(  
            tmp1.setColor(tmp2.getColor())  
            tmp1.BlockCell  
        tmp1 = tmp1.getNext()  
        tmp2 = tmp2.getNext()
```

Figura 12. Código InvertCell
Fuente: elaboración propia

Conclusiones

Al finalizar la solución al problema que se presentó se concluyó que la implementación de TDA a nuestros códigos puede ayudarnos a mejorar la memoria que utiliza nuestro programa y se pueden definir funciones específicas para que el programa sea más optimo

La implementación de POO en los algoritmos nos ayuda a almacenar ordenadamente nuestra información para que a la hora de extraer valores y operarlos no se confundan con los de otro objeto

La herramienta Graphviz nos ayuda a graficar cualquier matriz como nosotros queramos implementando el diseño que nos podamos imaginar

Referencias bibliográficas

- <https://rico-schmidt.name/pymotw-3/xml.etree.ElementTree/parse.html>
- <https://docs.python.org/3/library/xml.etree.elementtree.html>
- <https://www.geeksforgeeks.org/xml-parsing-python.html>