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.

PRIMER PROYECTO DE IPC 2

201902219 1 – Max Rodrigo Durán Canteo

Resumen

Creación de un robot capaz de convertir un patrón de azulejos de un piso a otro patrón tomando en cuenta el precio mínimo de voltear e intercambiar azulejos donde cada piso determina como un movimiento tiene cierto valor, la implementación de la lectura de datos se generó atravesó de un archivo xml leído y guardado en una lista enlazada propia con cada método echo desde cero. La implementación de la lista cuenta con agregar, eliminar, mostrar como texto, cambiar nodo, obtener nodo y mostrar su tamaño. Con el uso de una lista simple fue suficiente para la realización del programa, puesto que para la creación de matrices se introdujo una lista como nodo de una lista y así guardar los azulejos especificados por el patrón.

Palabras clave
Lista enlazada, Nodo, Matriz, Optimización.

Abstract

Creation of a robot capable of converting a pattern of tiles from one floor to another pattern taking into account the minimum price of flipping and exchanging tiles where each floor determines how a movement has a certain value, the implementation of the data reading was generated through an xml file read and saved in its own linked list with each method echoed from scratch. The list implementation features add, delete, display as text, change node, get node, and display its size. The use of a simple list was enough to carry out the program, since for the creation of matrices a list was introduced as a node of a list and thus save the tiles specified by the pattern.

Keywords
Linked List, Node, Matrix, Optimization.

Introducción

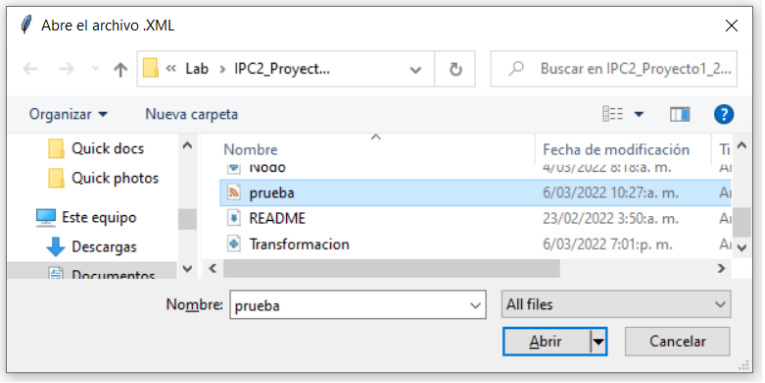
El programa lee un archivo xml el cual mediante la librería xml.etree.ElemtTree se va leyendo el diccionario de información, captando cada apartado en un lista enlazada, en cada piso se obtiene un costo de volteo, un costo de intercambio, tamaño de la matriz con n filas y m columnas donde se obtienen varios patrones, disco patrones son aceptados al programa solo al ser convertidos en una matriz cumplen con el tamaño establecido por el tipo de piso del que derivan, para así poder realizar el cambio de patrón de forma exitosa, posteriormente el programa ordena en orden alfabético cada código de patrón y cada nombre de piso, seguido del uso del programa, se empieza mostrando los diversos pisos donde el usuario puede ver la lista de pisos ordenados, seleccionar uno y se le mostrara el código de los patrones asignados a dicho piso, también ordenado, el usuario escogerá un código y podrá entre ver el patrón gráficamente o analizarlo para cambiar el patrón escogido por un nuevo.

Desarrollo del tema

En un principio el programa se desarrollará mostrando un menú inicial donde mostrará cuatro opciones en las cuales el usuario podrá escoger entre cargar datos, ver pisos, ver datos ordenados alfabéticamente y salir del programa.

```
<*>==================================<*>
 | 1. CARGAR DATOS                    |
 | 2. PISOS                           |
 | 3. DATOS EN FORMA ALFABETICA       |
 | 0. SALIR                           |
<*>==================================<*>
```

La opción cargar datos le pedirá al usuario un archivo con extensión xml y que cumpla con los parámetros del enunciado, mostrando así una ventana emergente en la cual aparecerá un cuadro grafico donde el usuario podrá escoger el archivo.



No obstante, si el archivo escogido no cumple con los parámetros establecidos, el programa obligara al usuario a que ingrese un archivo valido.

```
<*>==================================<*>
 | 1. CARGAR DATOS                    |
 | 2. PISOS                           |
 | 3. DATOS EN FORMA ALFABETICA       |
 | 0. SALIR                           |
<*>==================================<*>
1
[OPCION-CARGAR DATOS]
[ERROR]: Escoja un archivo valido
```

De forma que cuando sea valido el programa lo cargara y le indicara al usuario que fue cargado satisfactoriamente.

```
[CARGAR DATOS]: CARGA SATISFACTORIA
```

La carga de datos tiene un funcionamiento complejo, puesto que usa la librería xml, para su carga, se usa un solo método encargado de llevar la información a una variable global para su manejo de información.

En dicho método se va creando una serie de listas con nodos de tipo listas para el almacenamiento de datos y por cada vez que se termina de cargar n patrones de un piso, se usa un ordenamiento alfabético de tipo burbuja.

Teniendo de esta forma el método cargar datos:

```
def CargarDatos():
    key=True
    while key:
        try:
            tree = ET.parse(easygui.fileopenbox(title="Abre el archivo .XML"))
            root = tree.getroot()
            for i in range(len(root)):
                CurrentLista1=MiLista()
                CurrentLista1.Append(root[i].attrib["nombre"])
                CurrentLista1.Append(root[i][0].text.replace(" ","").replace("\n",""))
                CurrentLista1.Append(root[i][1].text.replace(" ","").replace("\n",""))
                CurrentLista1.Append(root[i][2].text.replace(" ","").replace("\n",""))
                CurrentLista1.Append(root[i][3].text.replace(" ","").replace("\n",""))
                CurrentLista2=MiLista()
                for j in range(len(root[i][4])):
                    if len(root[i][4][j].text.replace(" ","").replace("\n",""))==(int(
                        #print("aceptable")
                        CurrentLista3=MiLista()
                        CurrentLista3.Append(root[i][4][j].attrib["codigo"])
                        CurrentLista3.Append(root[i][4][j].text.replace(" ","").replac
                        CurrentLista3.Append(CrearMatrices(int(root[i][0].text.replace(
                        CurrentLista2.Append(CurrentLista3)
                Ordenamiento(CurrentLista2)
                CurrentLista1.Append(CurrentLista2)
                Data.Append(CurrentLista1)
            key=False
        except:
            print("[ERROR]: Escoja un archivo valido")
```

Las listas utilizadas fueron solamente una lista enlazada:

Clase Nodo
- Valor: Valor - SiguienteValor: None
+ __init__ + __str__

Clase ListaEnlazada
- Primero: None - Tamaño: int
+ __init__ + __str__ + Append(Valor) + Remove(Valor) + __setitem__(índice,NuevoValor) + __gettitem__(Indice) + __len__

Para el ordenamiento alfabético se utilizó un método con principios del ordenamiento burbuja donde se implementó un método externo para devolver un valor de tipo booleano identificando si el primero de los parámetros a evaluar es mayor o menor que el segundo parámetro, en caso de que sea mayor se retornara un True y el ordenamiento realizara el cambio. De lo contrario retornara un False y no se hará el cambio, solo se pasará a la siguiente iteración de la búsqueda.

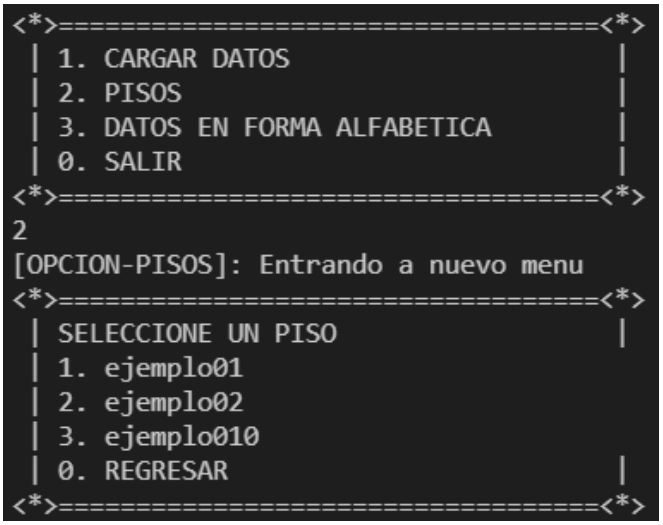
Siendo así el ordenamiento:

```
def Ordenamiento(Lista):
    for i in range(len(Lista)):
        for j in range(0, len(Lista)-i-1):
            primero=str(Lista[j][0])
            segundo=str(Lista[j+1][0])
            if esMayor(primero,segundo):
                temp = Lista[j]
                Lista[j] = Lista[j+1]
                Lista[j+1] = temp
```

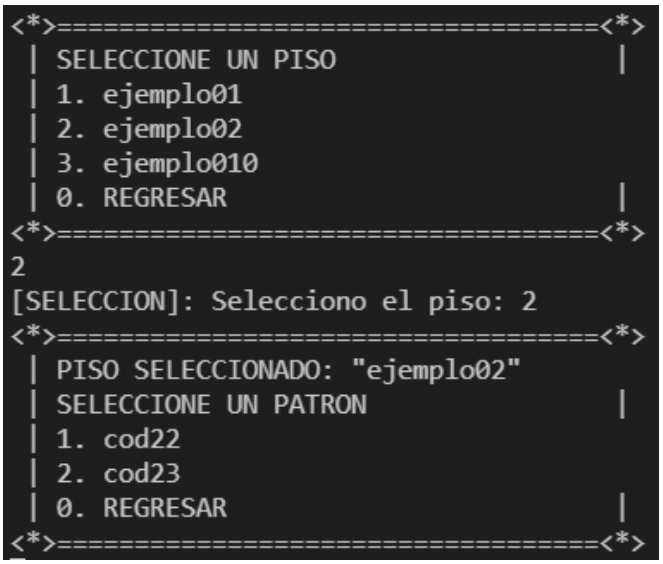
Siendo así el meto que define que parámetro es mayor:

```
def esMayor(primero,segundo):
    Lista1=MiLista()
    Lista2=MiLista()

    tamaño1=len(primero)
    tamaño2=len(segundo)

    for i in primero:
        Lista1.Append(i)
    for i in segundo:
        Lista2.Append(i)

    if tamaño1==1 and tamaño2>1:
        if ord(Lista1[0])>ord(Lista2[0]):
            return True
        elif ord(Lista1[0])==ord(Lista2[0]):
            return False
        else:
            return False
    elif tamaño1>1 and tamaño2==1:
        if ord(Lista1[0])>ord(Lista2[0]):
            return True
        elif ord(Lista1[0])==ord(Lista2[0]):
            return True
        else:
            return False
    elif tamaño1==tamaño2:
        for i in range(tamaño1):
            if ord(Lista1[i])>ord(Lista2[i]):
                return True
        return False
    elif tamaño1>tamaño2:
        for i in range(tamaño2):
            if ord(Lista1[i])>ord(Lista2[i]):
                return True
        return True
```

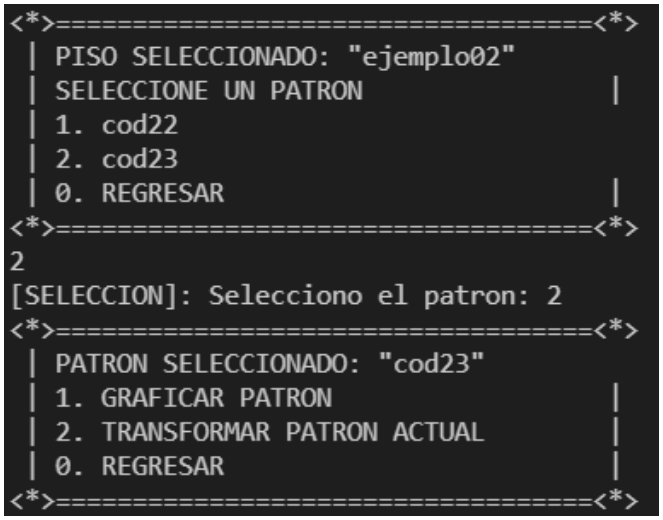
Pasando a la opción dos de nuestro menú principal, llegamos a la opción pisos, la cual mostrará la lista de pisos cargados en forma alfabética, el usuario podrá escoger un piso o regresar al menú principal.



En este apartado si el usuario escoge cualquier piso seguido por el número de índice que tiene a su izquierda, el programa le mostrara todos los patrones cargados a dicho piso.

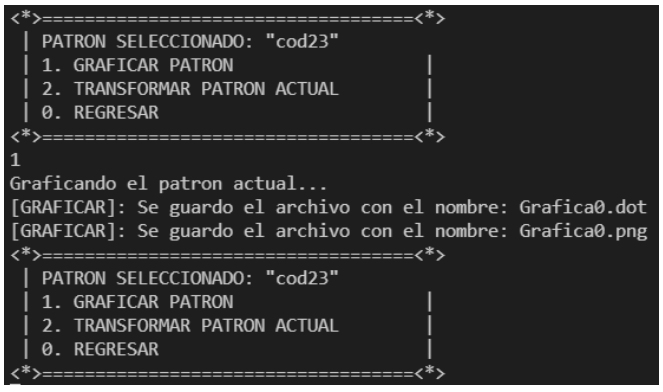


El usuario tendrá de nuevo dos opciones, escoger un patrón o regresar a la selección de pisos, si el usuario escoge un patrón de azulejo, lo llevara a un menú nuevo.

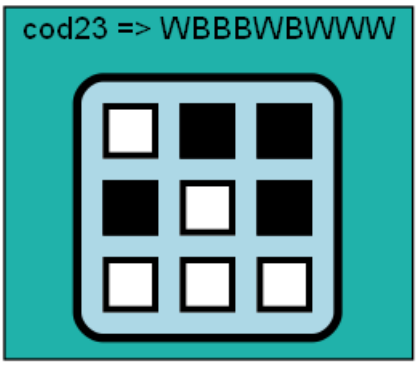


El usuario tendrá dos opciones nuevamente, graficar el patrón seleccionado o transformarlo en un nuevo o regresar al menú anterior.

Si el usuario selecciona graficar el patrono, se creará un archivo llamado GraficaN donde “N” será el índice del archivo en caso se generen más graficas y de extensión “DOT” el cual tendrá la información de lo que se graficara junto a su resultado en formato “PNG” llamado GraficaN donde “N” será el índice del archivo en caso se generen mas graficas. Y se mantendrá en el menú actual.



Para facilidad del usuario, el archivo png se abrirá automáticamente, mostrando la grafica del patrón de esta forma:



La grafica mostrara el código del patrón seguido del patrón cargado al sistema y el patrón de forma de matriz, siempre cumpliendo con las dimensiones tanto como del piso padre del que proviene.

La clase Graficar esta compuesta de esta manera:

Clase Graficar
- Matriz: Matriz - Codigo: Codigo + Patron - n: String
+ Crear + Calcular

Siguiendo con las opciones del menú anterior, si el usuario ingresa a “TRASNFORMAR PATRON ACTUAL” entrara a un menú donde tendrá que escoger un patrón del mismo piso padre, para convertir el primer patrón que escogió en este nuevo patrón.

```
<*>==================================<*>
 | PATRON SELECCIONADO: "cod23"
 | 1. GRAFICAR PATRON                 |
 | 2. TRANSFORMAR PATRON ACTUAL       |
 | 0. REGRESAR                        |
<*>==================================<*>
2
[SELECCION]: Selecciono transformar el patron actual
<*>==================================<*>
 | PISO SELECCIONADO: "ejemplo02"
 | SELECCIONE UN PATRON               |
 | 1. cod22
 | 2. cod23
 | 0. REGRESAR                        |
<*>==================================<*>
```

En esta opción, escogerá el patrón nuevo o podrá regresar al menú anterior.

```
<*>==================================<*>
 | 1. MOSTRAR COSTO MINIMO DEL CAMBIO  |
 | 2. MOSTRAR INSTRUCCIONES PASO A PASO|
 | 3. GRAFICAR NUEVO PATRON            |
 | 0. REGRESAR                         |
<*>==================================<*>
```

En el caso de escoger el patrón se tendrá un nuevo menú donde se podrá ver el costo mínimo por la transformación, las instrucciones, la gráfica del patrón.

En el caso del costo mínimo los mostrara de la siguiente forma:

```
El costo es de: 4 Q
```

En caso de requerir de las instrucciones se generará un menú donde el usuario podrá escoger entre mostrar las instrucciones en consola o mostrar las instrucciones en un archivo.

```
<*>==================================<*>
 | 1. MOSTRAR COSTO MINIMO DEL CAMBIO  |
 | 2. MOSTRAR INSTRUCCIONES PASO A PASO|
 | 3. GRAFICAR NUEVO PATRON            |
 | 0. REGRESAR                         |
<*>==================================<*>
2
<*>==================================<*>
 | 1. MOSTRAR INSTRUCCIONES EN CONSOLA |
 | 2. MOSTRAR INSTRUCCIONES UN ARCHIVO |
 | 0. REGRESAR                         |
<*>==================================<*>
```

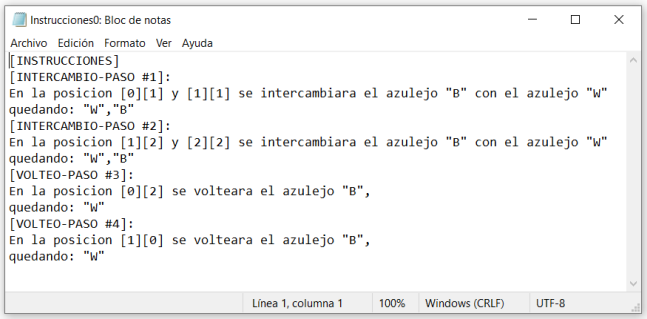
Si el usuario escoge mostrar las instrucciones en consola, se mostrará de la siguiente manera:

```
<*>================================<*>
 | 1. MOSTRAR INSTRUCCIONES EN CONSOLA |
 | 2. MOSTRAR INSTRUCCIONES UN ARCHIVO |
 | 0. REGRESAR                         |
<*>================================<*>
1
[INSTRUCCIONES]
[INTERCAMBIO-PASO #1]:
En la posicion [0][1] y [1][1] se intercambiara el azulejo "B" con el azulejo "W"
quedando: "W","B"
[INTERCAMBIO-PASO #2]:
En la posicion [1][2] y [2][2] se intercambiara el azulejo "B" con el azulejo "W"
quedando: "W","B"
[VOLTEO-PASO #3]:
En la posicion [0][2] se volteara el azulejo "B",
quedando: "W"
[VOLTEO-PASO #4]:
En la posicion [1][0] se volteara el azulejo "B",
quedando: "W"
```

Si el usuario escoge que quiere las instrucciones en un archivo, se abrirá automáticamente un archivo txt con las instrucciones para efectuar la transformación.

```
<*>================================<*>
 | 1. MOSTRAR INSTRUCCIONES EN CONSOLA |
 | 2. MOSTRAR INSTRUCCIONES UN ARCHIVO |
 | 0. REGRESAR                         |
<*>================================<*>
2
[INSTRUCCIONES]: Se guardo el archivo con el nombre: Instrucciones0.txt
<*>================================<*>
 | 1. MOSTRAR INSTRUCCIONES EN CONSOLA |
 | 2. MOSTRAR INSTRUCCIONES UN ARCHIVO |
 | 0. REGRESAR                         |
<*>================================<*>
```

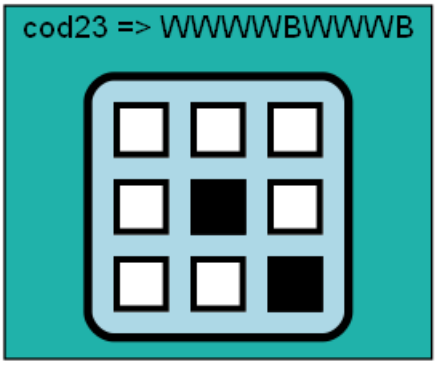
Mostrando el archivo txt de forma:



Si se regresa del menú para obtener las instrucciones, también podremos ver la gráfica del nuevo patrón:

```
<*>================================<*>
 | 1. MOSTRAR INSTRUCCIONES EN CONSOLA |
 | 2. MOSTRAR INSTRUCCIONES UN ARCHIVO |
 | 0. REGRESAR                         |
<*>================================<*>
0
[OPCION-REGRESAR]: Regresando
<*>================================<*>
 | 1. MOSTRAR COSTO MINIMO DEL CAMBIO  |
 | 2. MOSTRAR INSTRUCCIONES PASO A PASO|
 | 3. GRAFICAR NUEVO PATRON            |
 | 0. REGRESAR                         |
<*>================================<*>
3
Graficar nuevo patron
[GRAFICAR]: Se guardo el archivo con el nombre: Grafica1.dot
[GRAFICAR]: Se guardo el archivo con el nombre: Grafica1.png
<*>================================<*>
 | 1. MOSTRAR COSTO MINIMO DEL CAMBIO  |
 | 2. MOSTRAR INSTRUCCIONES PASO A PASO|
 | 3. GRAFICAR NUEVO PATRON            |
 | 0. REGRESAR                         |
<*>================================<*>
```

Mostrando la gráfica con el código del patrón ingresado por primera ver seguido de su transformación de forma léxica y grafica.



Dando así por concluido la transformación de patrones, también podremos regresar al menú principal y utilizar la opción “DATOS EN FORMA ALFABETICA”, donde se mostrará todos los pisos con sus respectivos patrones de azulejos en forma alfabética.

```
<*>==================================<*>
 | 1. CARGAR DATOS                   |
 | 2. PISOS                          |
 | 3. DATOS EN FORMA ALFABETICA      |
 | 0. SALIR                          |
<*>==================================<*>
3
[OPCION-DATOS EN FORMA ALFABETICA]: Mostrando datos
<*>==================================<*>
 | => Codigo de piso: ejemplo01
 |      II=> Codigo de patron: cod32
 |      II=> Codigo de patron: cod3131
 | => Codigo de piso: ejemplo02
 |      II=> Codigo de patron: cod22
 |      II=> Codigo de patron: cod23
 | => Codigo de piso: ejemplo010
 |      II=> Codigo de patron: cod11
 |      II=> Codigo de patron: cod12
<*>==================================<*>
[MENU]: Regresandoa menu principal
<*>==================================<*>
 | 1. CARGAR DATOS                   |
 | 2. PISOS                          |
 | 3. DATOS EN FORMA ALFABETICA      |
 | 0. SALIR                          |
<*>==================================<*>
```

Dando por concluido el programa.

Conclusiones

La utilización de listas enlazadas creadas desde cero, son mas funcionales que las listas por defecto de un lenguaje de programación.

La optimización con el uso de listas es complicada si no se usa una lista de tipo de pila.

Referencias bibliográficas

MÉNDEZ, Mariano. 75.41 Algoritmos y Programación II Tda Lista y sus Derivados.

OJEDA, Luis Roberto. Tda Programacion Orientado a Objetos en Turbo. Univ. Nacional de Colombia.

Extensión

Intercambio de pisos:

```
def ObtenerPrecios(self):
    #print(self.Piso)
    self.PrecioVolteo=self.Piso[3]
    self.PrecioIntercambio=self.Piso[4]

def AzulejosDesiguales(self):
    cont=0
    for i in range(self.filas):
        for j in range(self.columnas):
            if self.matriz2[i][j]!=self.matriz1[i][j]:
                #print("En la posicion ["+str(i)+"]["+str(j)+"] no coincide ")
                cont+=1
    return cont
```

```
def VolteoCoincidencia(self):
    for i in range(self.filas):
                for j in range(self.columnas):
                    if self.matriz2[i][j]!=self.matriz1[i][j]:
                        self.Pasos+=1
                        self.Instrucciones+="[VOLTEO-PASO #"+str(self.P
                        self.CostoMinimo+=int(self.PrecioVolteo)
                        self.matriz1[i][j]=self.matriz2[i][j]
```

```
def IntercambioLateralCoincidencia(self):
    for i in range(self.filas):
                for j in range(self.columnas):
                    if (j+1)<self.columnas:
                        if self.matriz2[i][j]!=self.matriz1[i][j] and self.matriz2[i][j+1]!=self

                            tengo1=self.matriz1[i][j]
                            tengo2=self.matriz1[i][j+1]
                            quiero1=self.matriz2[i][j]
                            quiero2=self.matriz2[i][j+1]

                            if tengo1=="B" and tengo2=="W" and quiero1=="W" and quiero2=="B":
                                self.Pasos+=1
                                self.Instrucciones+="[INTERCAMBIO-PASO #"+str(self.Pasos)+"]:\nE
                                self.CostoMinimo+=int(self.PrecioIntercambio)
                                temp=self.matriz1[i][j]
                                self.matriz1[i][j]=self.matriz1[i][j+1]
                                self.matriz1[i][j+1]=temp
                            elif tengo1=="W" and tengo2=="B" and quiero1=="B" and quiero2=="W":
                                self.Pasos+=1
                                self.Instrucciones+="[INTERCAMBIO-PASO #"+str(self.Pasos)+"]:\nE
                                self.CostoMinimo+=int(self.PrecioIntercambio)
                                temp=self.matriz1[i][j]
                                self.matriz1[i][j]=self.matriz1[i][j+1]
                                self.matriz1[i][j+1]=temp

def IntercambioVerticalCoincidencia(self):
```

```
def IntercambioVerticalCoincidencia(self):
    for i in range(self.filas):
                for j in range(self.columnas):
                    if (i+1)<self.filas:
                        if self.matriz2[i][j]!=self.matriz1[i][j] and self.matriz2[i+1][j]!=self.matriz1[i+1][j]:

                            tengo1=self.matriz1[i][j]
                            tengo2=self.matriz1[i+1][j]
                            quiero1=self.matriz2[i][j]
                            quiero2=self.matriz2[i+1][j]

                            if tengo1=="B" and tengo2=="W" and quiero1=="W" and quiero2=="B":
                                self.Pasos+=1
                                self.Instrucciones+="[INTERCAMBIO-PASO #"+str(self.Pasos)+"]:\nEn la posicion ["+str
                                self.CostoMinimo+=int(self.PrecioIntercambio)
                                temp=self.matriz1[i][j]
                                self.matriz1[i][j]=self.matriz1[i+1][j]
                                self.matriz1[i+1][j]=temp
                            elif tengo1=="W" and tengo2=="B" and quiero1=="B" and quiero2=="W":
                                self.Pasos+=1
                                self.Instrucciones+="[INTERCAMBIO-PASO #"+str(self.Pasos)+"]:\nEn la posicion ["+str
                                self.CostoMinimo+=int(self.PrecioIntercambio)
                                temp=self.matriz1[i][j]
                                self.matriz1[i][j]=self.matriz1[i+1][j]
                                self.matriz1[i+1][j]=temp
```