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.

MINIZACIÓN DE COSTOS PARA PROCESO DE CAMBIOS DE PATRONES

202006688 – Karla Ernestina González Polanco

Resumen

El primer proyecto del curso "Introducción a la Programación y Computación 2", tiene como objetivo, que el estudiante pueda comprender el proceso de manipulación que hace la computadora para algunas estructuras ya definidas dentro de un lenguaje de programación, implementando las suyas desde 0.

Para la implementación de estas estructuras determinadas como "TDA", se asignó al estudiante el análisis de un archivo con extensión XML, el cual contenía una lista de pisos, donde cada uno de ellos tenía sus características propias como:

- Número de columnas.
- Número de filas.
- Costo por acción de "flip".
- Costo por acción de "slide".
- Lista de patrones para cada piso.

El estudiante tiene que hacer su propia implementación de estructuras capaces de almacenar la información anteriormente descrita.

Además de ser capaz de manejarla para hacer un algoritmo de minimización de costos y graficación.

Palabras clave

TDA: Tipo de Dato Abstracto
Nodo: Registro que contiene datos y apuntadores.
XML: Lenguaje de marcado extensible.

Abstract

The first "Introducción a la Programación y computación 2" course project, has as purpose, allow the student to understand de manipulation process a computer must go through with the predefined structures a programming language has, by implementing his own structures right from the start.

For the implementation of these structures called "TDAs", the student was assigned to analyze an XML extension file, which contains a floor list, every floor has its own characteristics such as:

- *Number of columns.*
- *Number of rows.*
- *Cost for a "flip" action.*
- *Cost for a "slide" action.*
- *A patterns list for every floor.*

The student must do his own structures implementation, allowed to store the previously described information.

Also, having the capability to manage it in order to obtain a minimum cost algorithm and get a floor graph.

Keywords

ADT: Abstract Data Type

Node: Register which contains data and pointers.

XML: Extensible Markup Language

Introducción

Para el desarrollo del proyecto fue necesaria la implementación de TDAs, como listas simplemente enlazadas y doblemente enlazadas. En este caso se utilizaron tres tipos de nodos que almacenarían la información necesaria del archivo de entrada; nodoPiso, nodoPatron y nodoCelda. Cada uno de los anteriores destinado a ser los registros de la información que manejaría su lista respectiva.

El proyecto tiene como base dos acciones "principales": la de cambiar el patrón de un piso a un nuevo patrón que el usuario escoja utilizando el menor dinero posible y la de obtener su gráfica con un archivo tipo .jpg.

Las gráficas fueron creadas usando graphviz y adaptando su estructura para que pudiera ser vista gráficamente el patrón de un piso.

Desarrollo del tema

La empresa "Pisos Artesanales, S.A." ha construido un azulejo especial con el que puede crear pisos con distintos patrones. Cada piso consiste en una matriz de R filas y C columnas de azulejos cuadrados. Cada azulejo es reversible, un lado es blanco y el otro es negro, para poder crear patrones diversos. Además, la empresa garantiza que, para los pisos ya instalados, podrá cambiar el patrón original por un nuevo patrón que el cliente desee sin necesidad de comprar nuevos azulejos.

La empresa ha comprado un robot especializado capaz de colocar los pisos de dimensiones R x C con cualquier patrón, combinando azulejos del lado blanco con azulejos del lado negro. Además, la empresa garantiza que cada piso colocado podrá cambiar el patrón, a cualquier patrón deseado siguiendo las siguientes reglas: Para cambiar el patrón original del piso colocado por "Pisos Artesanales, S.A.", se debe cumplir que el nuevo patrón corresponda a las dimensiones R x C del piso colocado y el robot será capaz de realizar una de las siguientes operaciones con cada uno de los azulejos que componen el piso:

1. Voltear un azulejo, cambiando el color visible de blanco a negro o viceversa, y

2. Intercambiar dos azulejos adyacentes (horizontal o verticalmente, pero no en diagonal), sin voltear ninguno.

A la empresa "Pisos Artesanales, S.A." le resulta en un costo que el robot realice cada una de las operaciones antes mencionadas, de tal manera que, realizar una operación de volteo de azulejo cuesta F Quetzales, mientras que realizar una operación de intercambio de azulejos cuesta S Quetzales.

Debido a que la empresa posee una cuota fija, se solicita garantizar que la modificación de un patrón inicial a un patrón destino tenga el costo mínimo posible para optimizar el uso del robot.

Para poder determinar este algoritmo y el de graficación es necesario tener un archivo de entrada con entrada XML que tenga obligatoriamente la estructura del de la Figura 1. Donde se puede observar que cada piso tiene sus propios atributos de nombre, número de filas y columnas, costo de acción de slide y flip y una sublista con los patrones disponibles para ese piso (cada patrón debe tener un código que lo identifique).

El archivo de entrada puede contener n cantidad de pisos, teniendo cada uno las dimensiones que la empresa requiera, siendo denotado el número de filas por las etiquetas <R> y el número de columnas por la etiqueta <C>.

```
<?xml version="1.0"?>
<pisosArtesanales>
    <piso nombre="ejemplo01">
        <R> 4 </R>
        <C> 1 </C>
        <F> 1 </F>
        <S> 1 </S>
        <patrones>
            <patron codigo="cod11">WBWB</patron>
            <patron codigo="cod12">BWBW</patron>
        </patrones>
    </piso>
    <piso nombre="ejemplo02">
        <R> 1 </R>
        <C> 3 </C>
        <F> 1 </F>
        <S> 1 </S>
        <patrones>
            <patron codigo="cod21">WBB</patron>
            <patron codigo="cod22">WWW</patron>
        </patrones>
    </piso>
</pisosArtesanales>
```

Figura 1. Estructura de archivo de entrada.

Fuente: elaboración propia.

Para lograr el manejo de la información fue necesaria la implementación de tres listas, cada una con un tipo de nodo que represente el tipo de información que almacenará. Los nodos utilizados son entonces: nodoPiso, este almacenará la información general de cada piso: su nombre, número de filas, número de columnas, costo por acción de flip y costo por acción de slide. Además, este nodo tendrá una lista de patrones y una lista de celdas (esta última podría pensarse como una matriz pero representada de manera lineal). Tenemos también, el nodoPatron que contendrá el código y vagamente la cadena de texto que representa cada patrón de un piso.

Por último, el nodoCelda que contendrá la posición en x, la posición en y y el color que tendrá un azulejo de cada patrón de un piso.

Todos estos nodos son manejados a través de apuntadores, que harán referencia a los datos de sus nodos aledaños, en el sentido: anterior al nodo actual o siguiente al nodo actual.

Los nodos descritos anteriormente serán las estructuras de las cuales estarán formadas tres tipos de lista, siendo para el nodoPiso la listaPisos, para el nodoPatron la listaPatrones y para el nodoCelda la listaCeldas.

Estas listas son las que nos permitirán manejar la información para cada tipo de nodo. Siendo algunos de los métodos generales para una lista los siguientes:

a. InsertarAlFinal: este método inserta un nuevo nodo de cierto tipo a la lista en la que se requiera.
b. Mostrar: permite ver en consola la información que se le especifique para cada nodo contenido en cierta lista.
c. Buscar: hace uso de un parámetro el cual será buscado entre los atributos de cada nodo de cierta lista y en caso de encontrarlo, lo devolverá.

Además de los métodos generales existen otros más específicos que son utilizados solo para ciertos tipos de listas, los cuales son:

a) diferenteChar: recibe como parámetro un patrón de entrada y uno de destino. Está disponible para la lista de patrones y nos devolverá la posición en la que se encuentre un valor de color de azulejo diferente al comparar entre el patrón inicial y el patrón destino.
b) Reporte: disponible para la lista de celdas, este permite transferir la información de una lista lineal a una vista en forma de matriz, para que el usuario entienda de mejor manera cuál

es la estructura de un piso. Vea la figura 2, y figura 3. Para entender gráficamente la forma en la que funciona.

Figura 2. Vista lineal de la lista de celdas.

Fuente: elaboración propia.

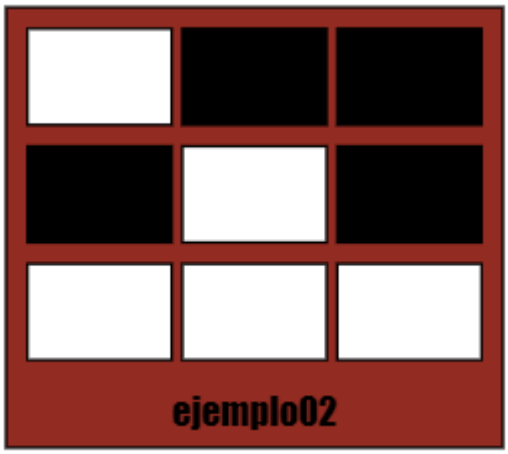

Figura 3. Vista como arreglo bidimensional de la lista de celdas.

Fuente: elaboración propia.

Por último, para calcular el costo mínimo para modificación del patrón inicial de un piso al patrón final, fue necesario primero determinar el número de cambios que tendrían que hacerse para poder llegar al patrón final. Este número podrá imprimirse en consola para tener una idea de la cantidad de cambios de azulejo que deberán hacerse.

A continuación, fue necesario determinar la posición en la que se encuentran las celdas que tendrán que ser cambiadas de color.

Primeramente, por cuestiones de optimización es necesario verificar si alguna de las celdas aledañas a la que necesita ser cambiada también tiene que ser cambiada, y si el nuevo color que debe tener es el color de esa celda actual, el menor precio será entonces el que se represente al hacer la acción de slide de las piezas. Esto significa que tendrá que ser necesario un manejo de los apuntadores para cambiar el valor. Esta acción se hará con el resto de las celdas hasta que todas las que estén posicionadas en alguna celda nueva puedan tener el valor correspondiente. A continuación, aquellas celdas que aún no tengan el color adecuado, y no tuvieron una acción de slide tendrán que ser cambiadas de color por descarte por medio de una acción de flip.

Conclusiones

La implementación de nuestras propias estructuras dinámicas nos da un mejor terreno para entender cómo es que funcionan las estructuras de los lenguajes de programación. Al ser estas estructuras predefinidas tan accesibles y “fáciles” de usar, podría perderse el razonamiento para el desarrollo de la solución a ciertos problemas que podrían presentarse en otras ocasiones.

También, su implementación nos permite tener las bases necesarias para poder hacer uso de ellas en cualquier otro lenguaje de programación, y sin ningún problema manejar información en forma de una lista, aunque este lenguaje tenga o no algo así entre sus estructuras.

El manejo de la información en archivos con extensión XML permite el aprendizaje de su manejo con diferentes tipos de lenguajes capaces de construir expresiones que puedan recorrerlo y de él extraer información importante para su posterior manejo en una aplicación.

Al hacer uso de graphviz, en su extensión .dot la implementación de sus diferentes características a una lista lineal de datos nos permite hacer diferentes cambias que nos permitan mostrar la información de

forma gráfica como algo mucho más entendible a como podría verse inicialmente.

Todos los conocimientos adquiridos con la realización del proyecto son piezas importantes para permitir el desarrollo de la lógica de implementación de nuevas estructuras y de hacer su correcto manejo para obtener los resultados necesarios al ejecutar su aplicación.

Referencias bibliográficas

Phyton TM, (2001-2022). *Minimal DOM implementation.* Python Software Foundation.

Emden R. Gansner and Eleftherios Koutsofios and Stephen North, (2015). *Drawing graphs whit dot.* Graphviz.

[Nombre del autor], (2006). *Tipos de datos abstractos (TDA).* Instituto tecnológico de Celaya.

Eduardo Zepeda (2021). Taller de graphviz completo.

Anexos

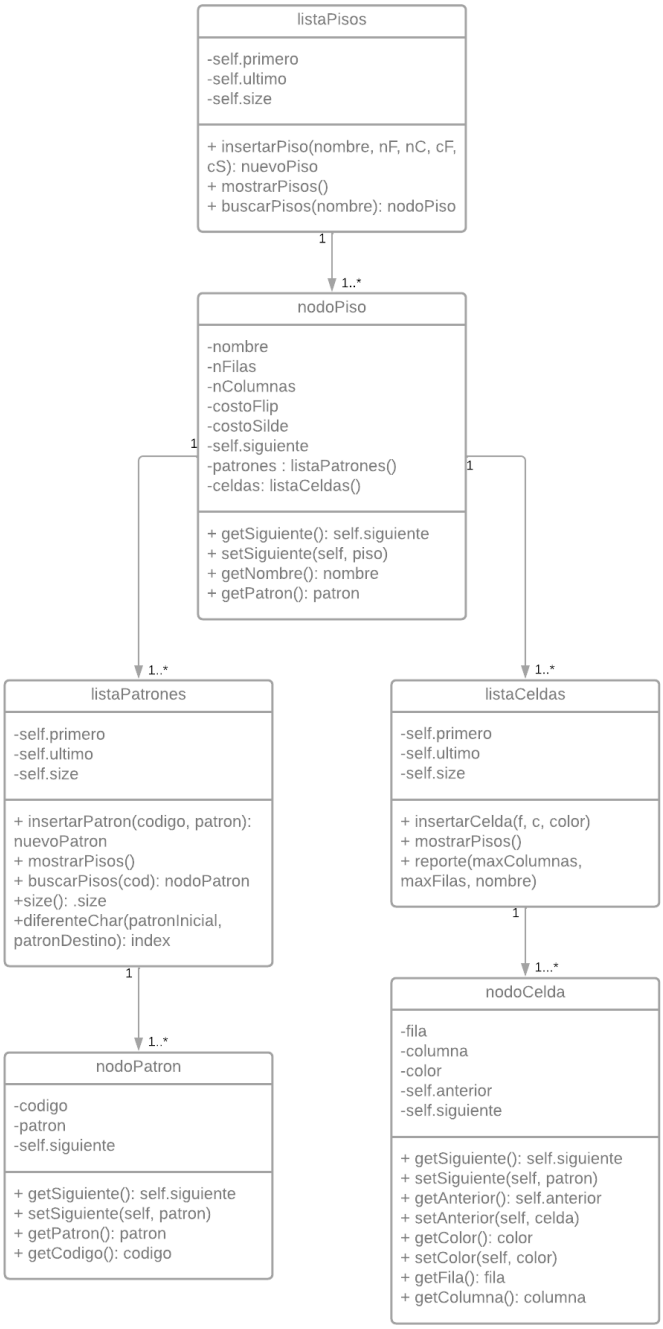

Figura 4. Diagrama UML.

Fuente: elaboración propia.