PROYECTO 2, MATRIZ ORTOGONAL

201801627 – Eduardo René Agustin Mendoza

Resumen

Teniendo en cuenta que dentro de una computadora hay espacios de memoria que cada uno de los componentes utiliza sabemos que en algún momento se van a consumir todos los recursos, para esto necesitamos utilizar ciertas herramientas para poder optimizar dicho espacio una de estas para tener en juego algunos espacios y poderlos utilizar es que se utilizaron objetos dinámicos esto para poder aprovechar estos espacios con esto se implementó una matriz dispersa con la cual se hará un espacio en memoria el cual estará solo designado para este objeto.

Utilizando los objetos dinámicos hacemos una optimización buena para la memoria y con esto podemos ocupar el espacio restante para alguna otra actividad

Palabras clave

Efectividad de almacenamiento, Matriz Dispersa, Funcionalidad, Movimiento, Juego

Abstract

Given that inside a computer there are spaces of memory that each component uses know that at some point they will consume all resources, for this we need to use certain tools to optimize the space one of these to take game some spaces and they can be used is that dynamic objects that were used to take advantage of these spaces with this a sparse matrix with which there will be a memory space which is only designated for this purpose was implemented

Using dynamic objects do good for memory optimization and with this we can occupy the remaining space for some other activity.

Keywords

Storage effectiveness, Dispersed Matrix, Functionality, Movement, Game.

Introducción

El desarrollo del programa está dentro del ámbito de programación de Python con la implementación de los TDA´s (Tipos de Datos Abstractos) esto para poder ir aprovechando el espacio en memoria que se va generando al ir guardando cada uno de los datos, no se utilizaron las listas predeterminadas de Python ya que su almacenamiento seria limitado y habría un desperdicio de memoria, a lo largo de esto se da una mejora al utilizar dichos TDA´s ya que se puede ingresar una infinidad de datos sin desperdiciar algún tipo de espacio ya que esto es vital en el proceso de los datos para no atrasarlo.

A lo largo del desarrollo de la aplicación se utilizo un archivo con estructura de XML puesto que dicha estructura simplificaba el trabajo a la hora de obtener los datos requeridos, al desarrollar la aplicación se pidió estrictamente que las matrices estuviesen guardadas en memoria y se realizaran las operaciones requeridas, para esto se creó una matriz dispersa (ortogonal) para poder realizar dichas operaciones para la realización de esto se tuvo que contar con apuntadores dentro de la memoria en donde se almacenarían los datos los cuales sería siguiente, anterior, arriba y abajo esto para poderse mover dentro de la matriz, también se conto con un acceso esto para poder adentrarse a la matriz desde la cabecera la cual es el nodo principal de cada fila y columna creada ya con esto definido se puede acceder a cualquier nodo interno de la matriz y empezar a mover la información a donde queramos o cambiarla con algunos datos, ya con esto se procede a insertar una matriz con posiciones y datos para iniciar la secuencia de las acciones.

Desarrollo del tema

La aplicación consiste en una forma de representar imágenes utilizando listas ortogonales y permitir realizar operaciones sobre estas imágenes. En la figura 1 se puede observar un ejemplo de una imagen almacenada en una matriz de caracteres.

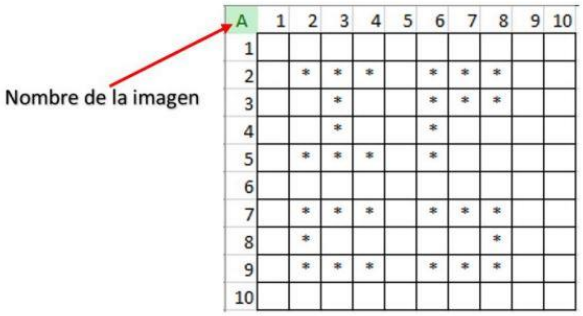

Figura 1. Imagen contenida en una matriz ortogonal de 10 filas por 10 columnas.

La figura 1 muestra una matriz denominada “A”, esta matriz es de 10 filas por 10 columnas y representa una imagen formada por el carácter “*” en los nodos que contienen información. Se requiere poder gestionar “N” imágenes en una lista simple ordenada. Estas imágenes tendrán un nombre y una dimensión. La dimensión de la imagen se determina por la cantidad de filas “f” y la cantidad de columnas “c” de esta imagen.

Estas operaciones se aplican sobre una imagen contenida en la lista ordenada de imágenes, modificando la imagen sobre la cual se aplican. Las operaciones que se podrán realizar sobre una imagen son las siguientes:

1. Rotación horizontal de una imagen
2. Rotación vertical de una imagen
3. Transpuesta de una imagen
4. Limpiar zona de una imagen
5. Agregar línea horizontal a una imagen
6. Agregar línea vertical a una imagen
7. Agregar rectángulo
8. Agregar triángulo rectángulo

Figura 2, Rotación Horizontal

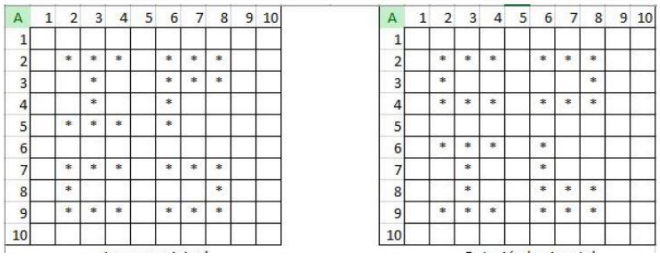

Figura 2. Ejemplo de Rotación Horizontal

Figura 3, Rotación Vertical.

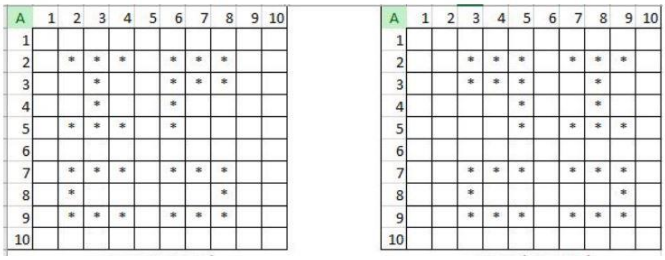

Figura 3. Ejemplo de Rotación Vertical

Figura 4, Transpuesta.

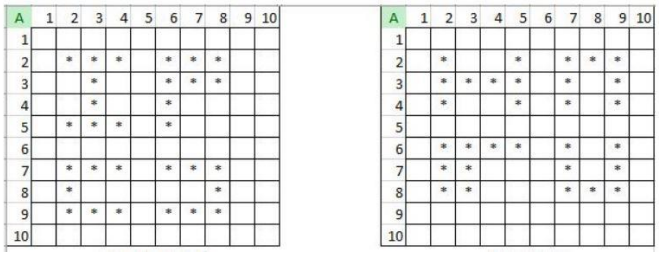

Figura 4. Ejemplo de Transpuesta

Figura 5, Limpiar Área

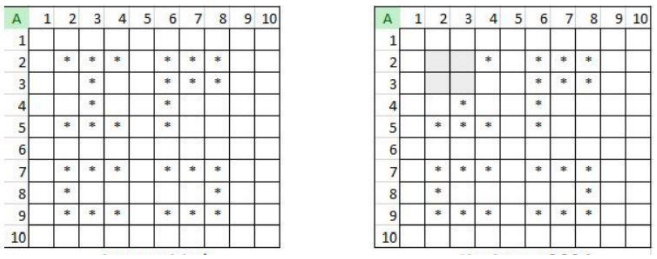

Figura 5. Ejemplo de limpiar zona

Figura 6, Agregar Línea Horizontal

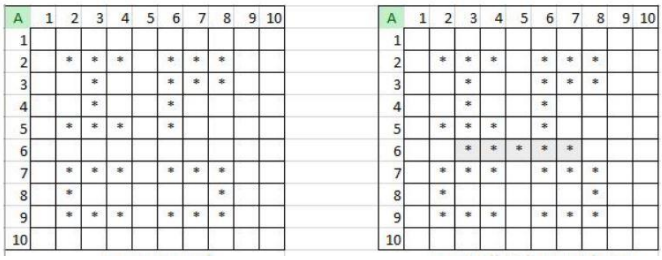

Figura 6. Agregar línea Horizontal

Figura 7, Agregar Línea Vertical

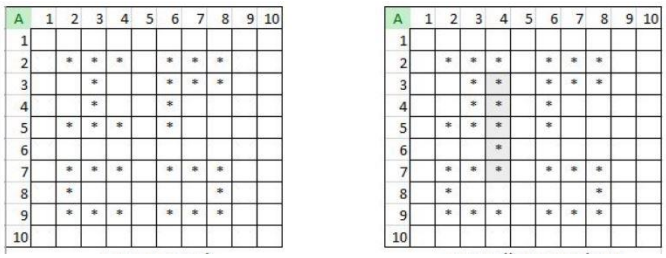

Figura 7. Agregar línea Vertical

Figura 8, Agregar un Rectángulo

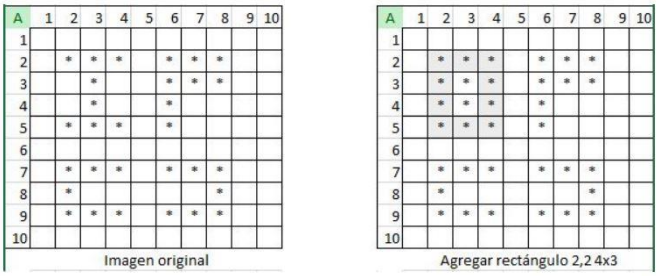

Figura 8. Agregar Rectángulo

Figura 9, Agregar Triángulo Rectángulo

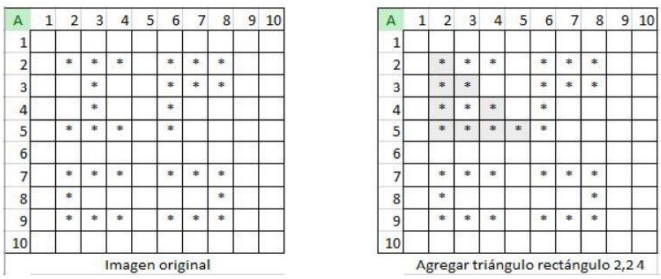

Figura 8. Agregar Rectángulo

Un TDA se define como una serie de n elementos en la cual se puede hacer cierta cantidad de operaciones, así como insertar, ver si esta vacía o no, eliminar

nodos, vaciar lista, mostrarla entre otras cada una de estas listas lleva consigo un nodo este contiene toda la información de lo que se ira guardando con un apuntador, dicho apuntador será único ya que dentro de la memoria solo existirá esa posición. Dentro de estos existen una cantidad especifica de listas en la cual entran los TDA cola (queue) y TDA pila (Stack) cada uno de estas estructuras tiene una simulación diferente por ejemplo una pila utiliza L.I.F.O = Last in, First out que sería último en entrar primero en salir esto lo podemos observar cuando tenemos apilados un par de libros que empezamos a extraer los libros de arriba hacía abajo y la cola F.I.F.O First in, First Out que sería primero en entrar y primero en salir esto podemos observarlo en una cola convencional de un banco que la primera persona que entra es la primera persona que sale, estos últimos tipos de datos son los que se utilizan más para almacenar un tipo de dato diferente ya que estos son para una larga duración entre otros.

La aplicación fue construida en el entorno de desarrollo de Python utilizando el IDE de Pycharm, la propuesta dada fue mejor el rendimiento del almacenamiento y minimizar los costos de trasmisión esto se dio para poder tener un esquema mejorado para que se adaptara al nuevo patrón que se almacenara.

La estructura de los datos se guardaran en archivos xml esto para poder tener un mejor despliegue de los datos ya que estos vendrán en una matriz los cuales se deben de trasformar en un código de entrada para poder ingresar a los servidores, el problema fue de que muchas de estos códigos se repetían por lo tanto la solución planteada fue que los códigos en este caso que eran binarios, si alguna fila de dicho documento se repetiría en su estructura, tengan el 100% de igualdad en el código binario, se irían sumando pero no el código si no al tener la matriz original comparar ambas líneas y verificar que sean las que se repiten entonces la suma de esa fila creara otro patrón de código el cual será una matriz reducida y este será mucho más fácil de poder ingresar a los servidores y ahorrara la lectura de filas repetidas, al momento de mostrar un dato de salida cada uno de los documentos que se dará tendrá la matriz reducida con la misma estructura xml.

Conclusiones

Sabemos que al trabajar con una extensa cantidad de datos es mucho mejor utilizar formatos de archivos que nos faciliten el trabajo y ahorren tiempo ya que el tiempo es muy valioso a la hora de verificar una gran cantidad de datos, con los códigos reducidos se agiliza este proceso.

A dicha solución del problema se implementaron las listas circulares y simples ya que en una circular iban a estar enlazadas todas las matrices que tienen los documentos y la simple fue nada más para almacenar los datos que traía cada una de las matrices para posteriormente analizarlas y comparar las filas si se repetían o no

Al trabajar con los TDA´s podemos ahorrarnos una infinidad de espacio ya que se utiliza lo necesario dentro del programa.

Las matrices que se plantean al final dan una mejor inserción a los servidores ya que se simplifica su código de acceso.

Referencias bibliográficas

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.