INFORME DE INGENIERIA

LABORATORIO UNIDAD 2

Presentado por:

Marisol Giraldo Cobo

Presentado al Profesor:

Andres Aristizábal

Universidad ICESI

Santiago de Cali, Septiembre 8 del 2019

**INFORME DE INGENIERÍA**

**CONTEXTO DEL PROBLEMA**

Minecraft es un videojuego creado por *Markus <<Notch>> Persson*, que nos permite desarrollar nuestros propios universos fantásticos y artísticos, mediante la colocación y destrucción de bloques.  Al ser un videojuego de mundo abierto, no tiene una misión concreta, solo consiste en la construcción libre mediante el uso de cubos con texturas tridimensionales. Los bloques representan distintos elementos de la naturaleza y el jugador puede desplazarse por su entorno y modificarlo mediante la creación, recolección y transporte de esos bloques.

**PASO 1: IDENTIFICACIÓN DEL PROBLEMA**

**IDENTIFICACIÓN DE NECESIDADES Y SINTOMAS:**

* Minecraft es un juego de mundo abierto, por lo que no tiene un objetivo específico.
* Minecraft posee un sistema de “crafteo” que les permite a los jugadores crear nuevos bloques a partir de otros correctamente ordenados mediante el uso de mesas de trabajo o desde una sección del inventario que posee la misma función, permitiendo al jugador tener una diversidad de bloques importante.
* Se ha detectado la referencia que hace cada casilla (que contiene un conjunto de bloques) en el inventario, hacia el contador interno de bloques del sistema, esto se da debido al uso de un algoritmo muy lento.
* Peticiones al área de experiencia de usuario para añadir features ​en la próxima versión para tener una modalidad nueva de acceso rápido y de construcción.
* Almacenar bloques del mismo tipo en la barra de acceso rápido, siendo estos bloques transportados directamente desde el inventario interno hacia esta barra de manera automática.
* La modalidad de construcción debería permitir el uso de n diferentes barras de acceso rápido, las cuales deberían variar al dar clic en una flecha al lado del menú de acceso rápido. De esta manera los jugadores incrementarían su productividad al momento

de construir.

**DEFINICIÓN DEL PROBLEMA**

El videojuego Minecraft requiere una aplicación que permita de una forma eficiente acceder a los bloques del sistema, e implementar nuevas características como las ​features ​de modalidad de acceso rápido y construcción, además de una visualización intuitiva para percibir los nuevos cambios del sistema y de esta forma atraer a nuevos usuarios al mundo de Minecraft.

**PASO 2: RECOPILACIÓN DE INFORMACIÓN**

*Estructura de Datos:*

Las estructuras de datos en programación son diferentes formas de organizar información para manipular, buscar e insertar estos datos de manera eficiente.

*Complejidad Temporal:*

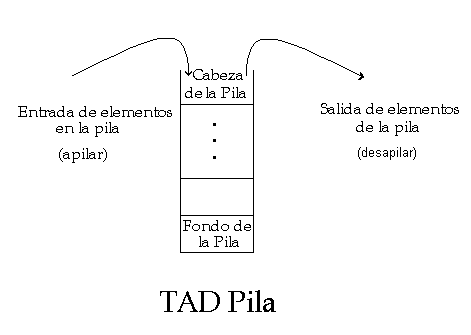
Se denomina complejidad temporal a la función T(n) que mide el número de instrucciones realizadas por el algoritmo para procesar los n elementos de entrada. Cada instrucción tiene asociado un costo temporal.

*Arreglos*

Los arrays son una estructura que almacena los datos un elemento al lado del otro. En la mayoría de los lenguajes de programación esta estructura de datos es de tamaño fijo y no puede guardar datos de diferentes tipos.

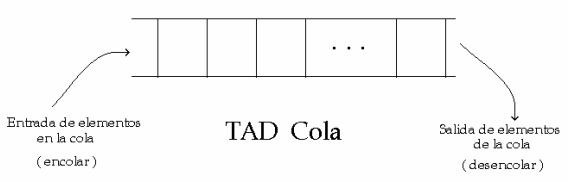
*Listas Enlazadas*

Las listas enlazadas son un tipo de estructura de datos similar a los arrays con la diferencia de que por defecto no tenemos por qué saber la cantidad de elementos que va a contener. Estas listas se componen de nodos los cuales tienen dos atributos: el primero es el ítem o elemento que va a contener este nodo y el segundo atributo es una referencia al siguiente elemento de la lista.



*Estructura de Datos - Lineales FIFO (Queue)*

Esta estructura es otro tipo de lista que nos permite emular el comportamiento de una fila o cola de la vida real donde el primer elemento en ingresar a la fila es el primero en salir, lo que quiere decir que las inserciones (Encolar) se realizan al final y las extracciones (Desencolar) se realizan al frente de la cola, lo cual se conoce como FIFO (First in First out).



*Estructura de Datos - Lineales LIFO (* Stacks)

Las pilas son un tipo de listas que tienen la particularidad de sólo poder eliminar o insertar en la cima de la lista. A estas acciones se le conocen como apilar y desapilar y conlleva a que el último elemento que ingresa a la pila sea el primero en salir a lo cual se le conoce como LIFO (Last in First out).

*Eficiencia Algorítmica*

Es usado para describir la cantidad de recursos utilizados por un algoritmo. Un algoritmo debe ser analizado para determinar el uso de los recursos que realiza.

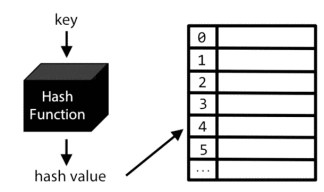
.

*Invariante:*

Es una condición que se sigue cumpliendo después de la ejecución de determinadas instrucciones. Se cumple tanto antes como después de estas instrucciones, permaneciendo sin variación.

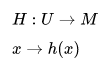
*Tabla de Hash:*

Es una estructura de datos que asocia *llaves* o *claves* con *valores*. La operación principal que soporta de manera eficiente es la *búsqueda*: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada (usando el nombre o número de cuenta, por ejemplo). Funciona transformando la clave con una **función hash** en un *hash*, un número que identifica la posición (*casilla* o *cubeta*) donde la tabla hash localiza el valor deseado.



*Función Hash*

Una función hash H es una función computable mediante un algoritmo tal que:

{\displaystyle H:U\rightarrow M}{\displaystyle x\rightarrow h(x)}

Tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte en un rango de salida finito, normalmente cadenas de longitud fija. Es decir, la función actúa como una proyección del conjunto U sobre el conjunto M. Hay que tener en cuenta que M puede ser un conjunto definido de enteros.

*TAD*

U n Tipo Abstracto de Datos es un conjunto de valores y de operaciones definidos mediante una especificación independiente de cualquier representación.

TAD = valores + operaciones

La manipulación de un TAD sólo depende de su especificación, nunca de su implementación

*Estructura de Datos – Diccionario*

En un diccionario de datos se encuentra la lista de todos los elementos que forman parte del flujo de datos de todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario de datos guarda los detalles y descripción de todos estos elementos, incluyendo nombre, descripción, alias, contenido y organización.

**PASO 3. BÚSQUEDA DE SOLUCIONES CREATIVAS**

**ALTERNATIVA 1: Stack (Pila)**

El acceso a los elementos de la pila se realizará siempre sobre un único extremo. Se insertará un nuevo elemento sobre la cima (**push**) y se extraerá el elemento situado en la cima (**pop**). Una forma de ver esta estructura de datos es como una pila de objetos en la que sólo se puede coger el objeto que está en la cima o apilar más objetos sobre la misma, pero los objetos que sostienen la pila no son accesibles pues de otro modo todo se desmoronaría.

Para este proceso se utilizará una función hash que se encargue de agregar los objetos a la pila usando como clave el nombre, la función hash colocará el primer objeto insertado en una fila respectiva de la tabla, y cada posición de la tabla hash tendrá una estructura de datos tipo pila que contenga como máximo de 64 objetos. En caso de que la pila este llena, ocurrirá una colisión y una segunda función hash la colocará en la posición al lado.

**ALTERNATIVA 2: Queue (Colas)**

Los elementos de la cola se añaden y se eliminan de tal manera que el primero en entrar es el primero en salir. La adición de elementos se realiza a través de la operación encolar (**enqueue**), mientras que la eliminación a través de la operación desencolar (**dequeue**). La operación de encolar inserta elementos por un extremo de la cola, mientras que la de desencolar los elimina por el otro.

Para este proceso se utilizará una función hash que se encargue de agregar los objetos a la cola usando como clave el nombre, la función hash colocará el primer objeto insertado en una fila respectiva de la tabla, y cada posición de la tabla hash tendrá una estructura de datos tipo cola que contenga como máximo de 64 objetos. En caso de que la cola este llena, ocurrirá una colisión y una segunda función hash la colocará en la posición al lado.

**Alternativa 3: Hash Table (Open Addressing)**

Mediante el direccionamiento abierto se podrá almacenar los registros directamente en el array. Las colisiones se resolverán mediante un *sondeo* del array, en el que se buscarán diferentes localidades del array hasta que el registro sea encontrado o si llega a una casilla vacía, indicando que no existe esa llave en la tabla. Este sondeo podrá resolverse de las siguientes maneras:

**1. Sondeo lineal**

En el que el intervalo entre cada intento es constante (frecuentemente 1).

**2. Sondeo cuadrático**

En el que el intervalo entre los intentos aumenta linealmente (por lo que los índices son descritos por una función cuadrática).

**3. Doble *hasheo***

En el que el intervalo entre intentos es constante para cada registro, pero es calculado por otra función hash. El doble hasheo tiene pobre rendimiento en el caché pero elimina el problema de aglomeramiento. Este puede requerir más cálculos que las otras formas de sondeo.

**PASO 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES**

**PASO 5: EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**PASO 6: PREPARACIÓN DE INFORMES Y ESPECIFICACIONES**

|  |  |  |
| --- | --- | --- |
|  | | Laboratorio I  Algoritmos y Estructuras de Datos  Requerimientos Funcionales |
| Nombre del Proyecto: | | Minecraft |
| Presentado por: | | Marisol Giraldo Cobo – Código: A00246380 |
| Presentado a: | | Andres Aristizábal |
| Fecha: | | Septiembre 15 del 2019 |
| Nombre | R1 – Adicionar bloques al inventario | |
| Resumen | Adicionar bloques a un conjunto | |
| Entradas | | |
| - typeOfBlocks :Representa el tipo de bloques  - blocks :Numero de bloques | | |
| Resultados | | |
| Los bloques son agregados a un slot del inventario | | |
| Nombre | R2 – Retirar los bloques del inventario | |
| Resumen | Remover los bloques que se encuentran en el inventario de acuerdo con la cantidad requerida | |
| Entradas | | |
| -typeOfBlocks: Representa el tipo de bloques  - blocks: Numero de bloques | | |
| Resultados | | |
| Los bloques son retirados del inventario | | |
| Nombre | R3 – Adicionar bloques al Quick-Access Bar | |
| Resumen | Adicionar bloques a la Quick -Access Bar | |
| Entradas | | |
| - typeOfBlocks :Representa el tipo de bloques | | |
| Resultados | | |
| Bloques agregados a la Quick-Access Bar | | |
| Nombre | R4 – Retirar bloques al Quick-Access Bar | |
| Resumen | Retirar bloques a la Quicsk-Access Bar | |
| Entradas | | |
| - typeOfBlocks :Representa el tipo de bloques | | |
| Resultados | | |
| Bloques retirados de la Quick-Access Bar | | |

**Paso 7: Implementación**

**BIBLIOGRAFÍA**

Minecraft

<https://www.minecraft.net/en-us/>

Estructura de Datos Tablas Hash

<https://cprog.cubava.cu/2018/03/25/tablas-hash/>