Complejidad temporal y espacial del algoritmo de ordenamiento:
Bubble Sort – Implementado para ordenar los juegos de cada estantería dependiendo del hash que arroja la tabla hash que corresponde a la estantería, para así saber cual juego se encuentra más abajo en esa estantería.

for(int i = 0; i < array.length; i++)	n
for(int j = 0; j < array.length-i-1; j++)	n^2
if(array[j] > array[j+1])	n^2
int temp = array[j];	n^2
array[j] = array[j+1];	n^2
array[j+1] = temp;	n^2

Este algoritmo tiene complejidad temporal de O(n^2).

Entrada	array	n
Salida	array	n
Auxiliar	i	1
Auxiliar	j	1
Auxiliar	temp	1

Este algoritmo tiene complejidad espacial de O(n).

Especificación de requerimientos funcionales

El programa debe estar en capacidad de ingresar todos los datos relevantes de la tienda: cajeros, estanterías, y por cada juego la cantidad y el precio. Después, el programa debe recibir los datos de cada cliente y preparar su orden, seleccionando la mejor ruta para buscar sus juegos deseados y mandándolos a las cajas. El programa debe procesar en el orden de llegada que por cada cajero disponible se atiende una persona y para cada turno en la caja se demora una unidad de tiempo (minuto) por cada juego en el carrito de compra y un turno adicional para el pago. Finalmente, se debe retornar el orden en que van saliendo los clientes, el cual puede ser distinto al orden en que se mandaron a las cajas por si uno con una orden mas pequeña logro pagar antes que otro que ya estaba, pero con una orden mas grande.

Requerimientos no funcionales

Para ordenar los juegos, se hace uso de un algoritmo de ordenamiento con complejidad de O(n^2), en este caso se ha utilizado el Bubble Sort. El programa va a hacer uso de una interfaz grafica para el ingreso y la salida de la información correspondiente. Los diferentes procesos del programa deben hacer uso de las estructuras de datos trabajadas en clase: colas, pilas y tablas hash.

TAD – Pila

Invariante: Todo objeto que se agrega siempre queda en la parte superior de la pila. Todo objeto que se remueve o consulta, es el objeto en la parte superior de la pila. Para toda pila vacía, top = null.

Operaciones primitivas:

Crear Pila Pila
Crea la pila.
Pre: true
Post: top = null

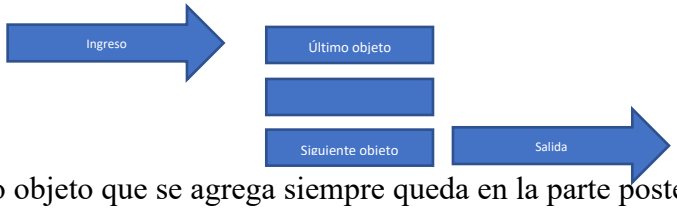
Empujar Objeto (push) Objeto booleano
Agrega un objeto a la parte superior de la pila, e indica si fue exitosa la operación.
Pre: Objeto a agregar
Post: top = Objeto

Remover Objeto (pop) Objeto
Remueve el objeto de la parte superior de la pila, y retorna el objeto.
Pre: top no es null
Post: Si stack.size > 1, top = Objeto siguiente, si no, top = null.

Consultar Objeto (peek) Objeto
Permite acceder al objeto que se encuentra en la parte superior de la pila, sin removerlo.
Pre: top no es null
Post: Objeto correspondiente a top

Esta Vacía booleano
Retorna si la pila esta vacía.
Pre: true
Post: Si top es null, true, si no, false

TAD – Cola



Invariante: Todo objeto que se agrega siempre queda en la parte posterior de la cola. Todo objeto que se remueve es el objeto en la parte frontal de la cola. Para toda cola vacía, front y rear = null.

Operaciones primitivas:

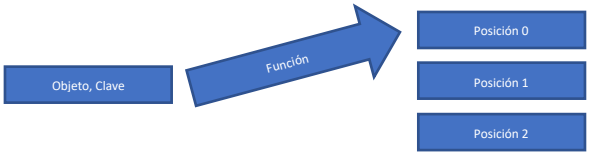
Crear Cola Cola
Crea la cola.
Pre: true
Post: front = null, rear = null

Agregar objeto (enqueue) Objeto booleano
Agrega un objeto a la parte posterior de la cola, e indica si fue exitosa la operación.
Pre: Objeto a agregar
Post: rear = Objeto, y si front = null, front = rear = Objeto

Remover Objeto (dequeue) Objeto
Remueve el objeto de la parte frontal de la cola, y retorna el objeto.
Pre: front no es null
Post: Si queue.size > 1, front = Objeto que sigue, si no front = rear = null

Esta Vacía booleano
Retorna si la cola esta vacía.
Pre: true
Post: Si front = null, true, si no, false

TAD – Tabla Hash



Invariante: Cada objeto obtiene su propia clave/código.

Operaciones primitivas:

Crear Tabla Hash entero Tabla Hash
Crea la tabla hash de tamaño determinado.
Pre: entero positivo
Post: T = T[entero]

Agregar objeto Objeto, clave booleano
Agrega un objeto a la tabla, en la posición que determine la función basada en la clave y la disponibilidad de espacio. Determina si fue exitosa la operación.
Pre: Objeto a agregar y su clave única
Post: T[función(clave)] = Objeto

Remover objeto Clave booleano
Remueve el objeto con dicha clave de la tabla, si existe. Determina si fue exitosa la operación.
Pre: Clave para remover
Post: T[función(clave)] = null

Buscar Objeto Clave Objeto
Busca el objeto con la clave indicada y lo retorna si lo encuentra.
Pre: Clave
Post: Objeto correspondiente a la clave, o null si no se encontró.

Esta Llena booleano
Retorna si la tabla hash esta llena.
Pre: true
Post: true si cada posición de T no es null, false de lo contrario.