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.