UNIVERSIDAD ICESI

Departamento de TIC
Computación y Estructuras Discretas I
2024-1 - Tarea Integradora 1

TEAM: VaLquiRia

- Martínez Vásquez Luna Catalina
- Mosquera Daza Renzo Fernando
- Tobar Gómez Valentina

Objetivos

Unidad 3: Estructuras Discretas no recursivas y análisis de algoritmos.

OE3.1 Explicar el significado de “mejor”, “promedio” y “peor” caso en lo que a comportamiento de algoritmos se refiere.

OE3.2 En el contexto de algoritmos específicos, identificar las características de los datos y/o otras condiciones o supuestos que conduzcan a distintos comportamientos.

OE3.3 Determinar, de manera informal, la complejidad temporal y espacial de algoritmos simples.

OE3.4 Establecer la definición formal de Big O.

OE3.5 Enumerar y contrastar las clases de complejidad de un algoritmo.

OE3.6 Calcular la complejidad temporal de algoritmos iterativos.

OE3.7 Calcular la complejidad espacial de algoritmos iterativos.

OE3.8. Diseñar e implementar un API para un proyecto de pequeña escala, utilizando un lenguaje de programación orientado a objetos, librerías e incluyendo parametrización y generics.

OE3.9. Diseñar e implementar una solución a un problema utilizando un lenguaje de programación teniendo en cuenta un criterio de eficiencia computacional y estándares de codificación seguros.

OE3.10 Definir, implementar y utilizar tablas hash, incluyendo técnicas de resolución de colisiones.

OE3.12 Definir, implementar y utilizar estructuras discretas FIFO y LIFO.

OE3.13 Desarrollar las pruebas unitarias para cada una de las estructuras discretas implementadas.

Unidad 4: Algoritmos y estructuras discretas recursivas

OE4.5 Definir, implementar y utilizar montículos binarios.

OE4.6 Describir la propiedad de montículos y el uso de estos como implementación de una cola de prioridad.

UNIVERSIDAD ICESI

Departamento de TIC
Computación y Estructuras Discretas I
2024-1 - Tarea Integradora 1

Descripción:

En este proyecto, se implementará un juego de cartas en Java que utilizará pilas, colas, tablas hash y colas de prioridad para gestionar diferentes aspectos del juego. El juego seleccionado para este proyecto será "Uno", un juego de cartas de colores en el que los jugadores intentan quedarse sin cartas en la mano.

Reglas del Juego:

1. Distribución de Cartas: se distribuirán 7 cartas a cada jugador desde una baraja estándar de 108 cartas.
2. Inicio del Juego: se coloca una carta boca arriba en el centro como carta inicial.
3. Turno del Jugador:
    a. Cada jugador, en su turno, debe jugar una carta que coincida en color, número o símbolo con la carta superior en el montón de descarte.
    b. Si un jugador no puede jugar ninguna carta, debe robar una carta del mazo. Si la carta robada puede ser jugada, el jugador la puede jugar inmediatamente; de lo contrario, se pasa al siguiente jugador.
4. Objetivo: el objetivo es quedarse sin cartas en la mano. El primer jugador en quedarse sin cartas gana la partida.
5. Cartas Especiales: las cartas especiales incluyen:
    a. Cambio de Color: permite al jugador cambiar el color del juego.
    b. Roba 2: el siguiente jugador debe robar 2 cartas y perder su turno.
    c. Revertir: invierte el orden de juego.
    d. Salto: hace que el siguiente jugador pierda su turno.

Estructuras de Datos Utilizadas:

- Pilas: se utilizan para representar el mazo de juego y el mazo de descarte.
- Colas: sirven para representar el mazo de cartas de los jugadores, donde se almacenan las cartas no jugadas y se roban nuevas cartas. Las cartas se distribuyen a los jugadores desde una cola y las cartas que se deben robar se agregan a la cola. Por ejemplo, cuando un jugador necesita robar una carta, se toma la primera carta de la cola.
- Tablas Hash: se usan para almacenar información adicional sobre las cartas, como su valor y su tipo (normal o especial). Para no tener que guardar en las colas y pilas el objeto carta sino una referencia, que luego se utilizará para acceder a dicha carta en la tabla hash.
- Colas de Prioridad: se utilizan para determinar el orden de juego cuando se usan cartas especiales como "Robar 2" o "Salto". Cuando se juega una carta especial, se necesita cambiar el orden en el que los jugadores están jugando. Por ejemplo, si un jugador juega una carta "Robar 2", el siguiente jugador que debería jugar sería el que está después del siguiente jugador (puesto que el siguiente jugador tiene que robar cartas). Esto requiere una reorganización del orden de los jugadores, lo que se puede lograr utilizando una cola de prioridad donde cada elemento tiene una prioridad asociada (en este caso, la posición del jugador).

Paso a Paso del Desarrollo:

1. **Análisis y Diseño:**
    a. Analizar los requisitos del juego y diseñar la arquitectura del programa, identificando las estructuras de datos necesarias.
2. **Implementación de las Estructuras de Datos:**
    a. Implementar las pilas, colas, tablas hash y colas de prioridad en Java.
3. **Distribución de Cartas:**
    a. Crear el método para distribuir las cartas a los jugadores y configurar el mazo de descarte.
4. **Turno del Jugador:**
    a. Implementar la lógica para permitir que los jugadores jueguen cartas y roben nuevas cartas si es necesario.
5. **Verificación de Reglas:**
    a. Validar que los movimientos de los jugadores cumplan con las reglas del juego y actualizar el estado del juego en consecuencia.
6. **Finalización del Juego:**
    a. Verificar si un jugador ha quedado sin cartas en la mano para determinar el ganador de la partida.
7. **Interfaz de Usuario (UI):**
    a. Crear una interfaz de usuario que permita a los jugadores interactuar con el juego, mostrando las cartas en la mano de cada jugador y las opciones disponibles en cada turno.
8. **Pruebas:**
    a. Realizar pruebas exhaustivas del juego para garantizar su correcto funcionamiento.

UNIVERSIDAD ICESI

Departamento de TIC
Computación y Estructuras Discretas I
2024-1 - Tarea Integradora 1

Entregables

1. Desarrollo completo del Método de la Ingeniería. (Ejemplo)
    a. La fase 1 debe incluir la especificación de requerimientos.
2. Análisis.
    a. Análisis de complejidad temporal de al menos dos de los algoritmos implementados.
    b. Análisis de complejidad espacial de al menos dos de los algoritmos implementados.
3. Diseño.
    a. Diseño del TAD de las estructuras de datos utilizadas.
    b. Diseño completo del diagrama de clases usando genéricos, incluyendo, las estructuras de datos, el paquete del modelo, de la interfaz de usuario y pruebas.
    c. Diseño de casos de prueba, incluyendo los escenarios, para las estructuras y el sistema.
4. Implementación en Java.
    a. Implementación completa de las estructuras de datos y sus pruebas.
    b. Implementación completa y correcta del modelo, la UI y las pruebas (contexto del problema).

Link del repositorio: [link]

Nota: La rúbrica con la que se evaluará esta tarea se encuentra en la pestaña TI1 de Notas y Seguimientos de su grupo. Se recomienda revisar la rúbrica con la que será evaluada su entrega.

Fecha Máxima de Entrega: 7 de Abril del 2024
