C:\Users\renzi\Desktop\playCard.java - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Complementos Pestañas ?

playCard.java startGame.java

```java
public boolean playCard(int cardIndex) {
    boolean flag = false; // O(1)

    Player currentPlayer = playerQueue.peek(); // O(1)
    String cardId = currentPlayer.getHand().get(cardIndex); // O(1)
    Card playedCard = deck.getCardTable().get(cardId); // O(1)

    String topCardId = deck.getPlayDeck().peek(); // O(1)
    Card topCard = deck.getCardTable().get(topCardId); // O(1)

    if (!changeColor && !changeColorController) { // O(1)
        auxiliaryCard = topCard.getColor(); // O(1)
    }

    boolean isChangeCard = playedCard.getSpecialType() == Card.SpecialType.CHANGE; // O(1)

    boolean canPlayNormalCard = playedCard.getSpecialType() == Card.SpecialType.NONE &&
            (playedCard.getColor() == auxiliaryCard || playedCard.getNumber() == topCard.getNumber()); // O(1)

    boolean canPlaySpecialCard = playedCard.getSpecialType() != Card.SpecialType.NONE &&
            (playedCard.getColor() == auxiliaryCard ||
                    playedCard.getSpecialType() == topCard.getSpecialType()); // O(1)

    if (isChangeCard || canPlayNormalCard || canPlaySpecialCard) { // O(1)
        currentPlayer.removeCardFromHand(cardId); // O(1) + O(n)
        deck.getPlayDeck().push(cardId); // O(1)

        gameOver = checkGameOver(); // O(1) + O(8) = O(9)
        flag = true; // O(1)
        changeColorController = false; // O(1)

        if (playedCard.getSpecialType() != Card.SpecialType.NONE) { // O(1)
            activeSpecialcard = true; // O(1)
            if (isChangeCard) { // O(1)
                changeColor = true; // O(1)
            }
        }
        nextTurn(); // O(1) + O(n)
    }
    return flag; // O(1)
}

//O(20) + O(n) + O(n)
//O(n)
```

Java source file | length : 1.722 lines : 45 | Ln : 45 Col : 1 Pos : 1.723 | Windows (CR LF) | UTF-8 | INS

C:\Users\renzi\Desktop\startGame.java - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Complementos Pestañas ?

playCard.java | startGame.java

```java
public void startGame(List<String> playerNames) {
    List<Player> players = new ArrayList<>(); // O(1)

    int priority = 1; // O(1)

    for (String name : playerNames) { //O(n+1)  == O(n)
        Player player = new Player(name); // O(1) * n  = O(n)
        players.add(player); // O(1) * n  = O(n)
        playerQueue.enqueue(player, priority); // O(1) * n  = O(n)
        priority++; // O(1) * n  = O(n)
    }

    Queue<String> cardsToDeal = new Queue<>(); // O(1)
    int totalCardsToDeal = players.size() * 7; // O(1)

    for (int i = 0; i < totalCardsToDeal; i++) { //O(n+1)   == O(n)
        cardsToDeal.enqueue(deck.getDiscardDeck().pop()); // O(1) * n = O(n)
    }

    // O(c), c es el número total de cartas a repartir
    // O(p), p es el número de jugadores
    while (!cardsToDeal.isEmpty()) { // O(c)

        for (Player player : players) { // O(p)
            if (!cardsToDeal.isEmpty()) { //O(c/p) --> en el peor de los casos
                player.addCardToHand(cardsToDeal.dequeue()); //O(c/p) --> en el peor de los casos
            }
        }
    }
    //Para este caso del while que tiene un for dentro, la complejidad temporal
    //de este es O(c), ya que cada carta se reparte una vez.

    while (!deck.getDiscardDeck().isEmpty()) { // O(m)
        String cardId = deck.getDiscardDeck().peek(); // O(1) * m  = O(m)
        Card card = deck.getCardTable().get(cardId); // O(1) * m  = O(m)

        if (card.getSpecialType() == Card.SpecialType.NONE) { // O(1) * m  = O(m)
            deck.getPlayDeck().push(deck.getDiscardDeck().pop()); // O(1) * m  = O(m)
            break; // O(1),
        } else {
            deck.getPlayDeck().push(deck.getDiscardDeck().pop()); // O(1) * m = O(m)
        }
    }
}

//O(MAX(n, c, m))
```

Java source file | length : 1.805 lines : 47 | Ln : 47 Col : 1 Pos : 1.806 | Windows (CR LF) | UTF-8 | INS

COMPLEJIDAD ESPACIAL DE LOS ALGORITMOS DE playCard() Y startGame()

playCard(int cardIndex)		
Tipo	**Variable**	**Cantidad de valores atómicos**
Entrada	cardIndex	1
Auxiliar	currentPlayer	1
Auxiliar	cardId	1
Auxiliar	playedCard	1
Auxiliar	topCardId	1
Auxiliar	topCard	1
Auxiliar	auxiliaryCard	1
Auxiliar	isChangeCard	1
Auxiliar	canPlayNormalCard	1
Auxiliar	canPlaySpecialCard	1
Auxiliar	gameOver	1
Auxiliar	changeColorController	1
Auxiliar	activeSpecialcard	1
Auxiliar	changeColor	1
Salida	flag	1

Complejidad Espacial Total = Entrada + Auxiliar + Salida
Entrada =1, + Auxiliar = 13, + Salida = 1
Complejidad Espacial Total = 15
15 == 15n^0
n^0 == 1
O(1)

startGame(List<String> playerNames)		
Tipo	**Variable**	**Cantidad de valores atómicos**
Entrada	playerNames	n
Auxiliar	players	n
Auxiliar	player	n
Auxiliar	priority	1
Auxiliar	cardsToDeal	7n
Auxiliar	totalCardsToDeal	1
Auxiliar	cardId	1
Auxiliar	card	1
Auxiliar	playerQueue	1
Auxiliar	deck	1

Complejidad Espacial Total = Entrada + Auxiliar + Salida
Entrada =n, + Auxiliar = 9n + 6, + Salida = 0
Complejidad Espacial Total = 10n + 6
10n == 10n^1
n^1 == n
O(n)