Minesweeper.java

**1. Các cấu trúc dữ liệu sử dụng (Data Structures)**

| **Tên trong mã gốc** | **Kiểu dữ liệu** | **Mục đích sử dụng** |
| --- | --- | --- |
| MineTile[][] board | Mảng 2 chiều của JButton | Đại diện cho bảng chơi Minesweeper, mỗi ô là một nút có thể click. |
| ArrayList<MineTile> mineList | Danh sách động (ArrayList) | Lưu vị trí các ô có mìn đã được gán. |
| Random random | Đối tượng Random | Sinh số ngẫu nhiên để gán mìn. |

**📌 2. Các thuật toán chính đã triển khai (Algorithms Implemented)**

**➤ setMines()**

* **Chức năng:** Gán mineCount mìn ngẫu nhiên vào các ô trên bảng.
* **Cách hoạt động:** Sinh ngẫu nhiên tọa độ (r, c) và kiểm tra nếu chưa có mìn thì thêm vào mineList.
* **Độ phức tạp thời gian:**
  + Trường hợp trung bình: **O(m)** với m = mineCount, vì danh sách mìn kiểm tra bằng contains() có độ phức tạp O(n) (n = số mìn đã gán), nên **O(m²)** trung bình.
  + Tệ nhất: Gần **O(n \* m)** nếu sinh trùng nhiều (m gần tổng số ô).

**➤ checkMine(int r, int c)**

* **Chức năng:** Kiểm tra xem ô (r, c) có mìn xung quanh không, nếu không thì đệ quy kiểm tra các ô xung quanh tiếp.
* **Thuật toán:** DFS đơn giản bằng gọi đệ quy (flood fill).
* **Độ phức tạp thời gian:**
  + Tốt nhất: **O(1)** (ô có số mìn lân cận > 0).
  + Trung bình/tệ nhất: **O(n)**, với n = numRows \* numCols, vì có thể kiểm tra hết bảng nếu không có mìn gần.

**➤ revealMines()**

* **Chức năng:** Khi click trúng mìn, hiển thị toàn bộ mìn đã gán.
* **Thuật toán:** Duyệt toàn bộ mineList.
* **Độ phức tạp thời gian:** **O(m)** với m = số mìn.

**➤ countMine(int r, int c)**

* **Chức năng:** Đếm xem ô (r, c) có phải là mìn không.
* **Thuật toán:** Kiểm tra xem mineList.contains(board[r][c]).
* **Độ phức tạp thời gian:** **O(m)** vì .contains() trên ArrayList là tuyến tính.

**📌 3. Tổng kết độ phức tạp**

| **Thuật toán** | **Độ phức tạp thời gian** |
| --- | --- |
| Khởi tạo bảng initializeBoard() | O(n) với n = hàng × cột |
| Đặt mìn setMines() | O(m²) (trung bình) |
| Kiểm tra mìn checkMine() | O(n) (đệ quy toàn bảng) |
| Hiển thị mìn revealMines() | O(m) |
| Đếm mìn countMine() | O(m) |

App,java

**1. Cấu trúc dữ liệu thực sự được sử dụng trong App.java**

| **Tên biến trong mã** | **Kiểu dữ liệu** | **Vai trò** |
| --- | --- | --- |
| homeFrame | JFrame | Cửa sổ chính hiển thị màn hình chào ban đầu. |
| homePanel | JPanel | Bảng chứa các thành phần giao diện màn hình chính. |
| boomLabel, welcomeLabel, instructionsLabel | JLabel | Hiển thị biểu tượng 💣, tiêu đề và hướng dẫn người chơi. |
| playButton | JButton | Nút "Play now" dùng để chuyển sang màn hình chọn cấp độ. |
| levelFrame | JFrame | Cửa sổ mới để chọn độ khó (easy, medium, hard). |
| levelPanel | JPanel | Bảng chứa giao diện màn hình chọn cấp độ. |
| easyButton, mediumButton, hardButton | JButton | Các nút chọn độ khó, khởi tạo Minesweeper với số mìn tương ứng. |

**📌 2. Các hàm & thuật toán được cài đặt**

**▶ public static void main(String[] args)**

* **Gọi showHomeScreen()** để hiển thị giao diện chào đầu.

**▶ showHomeScreen()**

* **Chức năng:** Tạo cửa sổ chào với:
  + Biểu tượng 💣
  + Tiêu đề "Enter the Boom Field!"
  + Hướng dẫn "Click to explore..."
  + Nút playButton chuyển sang showLevelScreen()
* **Thuật toán xử lý sự kiện:**
  + Khi nhấn playButton, gọi:

java

Sao chépChỉnh sửa

homeFrame.dispose(); // Đóng cửa sổ chào

showLevelScreen(); // Hiển thị giao diện chọn độ khó

* **Độ phức tạp thời gian:** O(1) — khởi tạo giao diện, không xử lý logic trò chơi.

**▶ showLevelScreen()**

* **Chức năng:** Hiển thị cửa sổ mới để người chơi chọn cấp độ.
* Các nút:
  + easyButton → new Minesweeper(10)
  + mediumButton → new Minesweeper(20)
  + hardButton → new Minesweeper(30)
* **Thuật toán xử lý sự kiện:**

java

Sao chépChỉnh sửa

levelFrame.dispose(); // Đóng cửa sổ chọn cấp độ

new Minesweeper(X); // Mở trò chơi mới với X mìn

* **Độ phức tạp thời gian:**
  + Xử lý sự kiện GUI: O(1)
  + Gọi new Minesweeper(X) → phức tạp phụ thuộc vào số mìn (xem bên dưới).

**📌 3. Giao tiếp với Minesweeper.java**

Khi người dùng chọn mức độ, App.java gọi:

java

Sao chépChỉnh sửa

new Minesweeper(10); // hoặc 20 hoặc 30

Trong Minesweeper.java, constructor sẽ khởi tạo lưới MineTile[][], đặt ngẫu nhiên mìn, đếm số mìn gần kề.

→ Đây là nơi có các thuật toán có độ phức tạp cao hơn:

* Khởi tạo lưới: O(n) (với n = số ô)
* Đặt mìn: O(m²) (với m = số mìn)

**📊 Tổng kết độ phức tạp thời gian trong App.java**

| **Hành động** | **Độ phức tạp** |
| --- | --- |
| Hiển thị homeFrame | O(1) |
| Nhấn nút Play now | O(1) |
| Hiển thị levelFrame | O(1) |
| Nhấn nút Easy/Medium/Hard | O(n + m²) → từ new Minesweeper(x) |

Update Time complexity

| **Class** | **Method** | **Time Complexity** | **Đã đúng?** |
| --- | --- | --- | --- |
| Minesweeper | Minesweeper() | O(n²) | ✅ |
| Minesweeper | initializeBoard() | O(n²) | ✅ |
| Minesweeper | placeMines() | O(m) | ✅ |
| Minesweeper | calculateAdjMines() | O(n²) | ✅ |
| Minesweeper | countAdjacentMines() | O(1) | ✅ |
| Minesweeper | revealCell() | O(n²) (worst) | ✅ |
| Minesweeper | isWin() | O(n²) | ✅ |
| Minesweeper | getCell() | O(1) | ✅ |
| Minesweeper | getMines() | O(1) | ✅ |
| App | main() | O(n²) | ✅ |
| App | printBoard() | O(n²) | ✅ |