

**合肥学院**

**人工智能与大数据学院**

**课程设计报告**

**2023 - 2024学年第 二 学期**

|  |  |
| --- | --- |
| **课程(综合)设计名称** | 程序与算法综合设计 |
| **专业班级** | 软工2班 |
| **姓名（学号）** | 钟健（22302092022） |
| **同组成员（学号）** | 徐杰东（22302092033） |
|  |  |
| **评阅教师** | 华珊珊 |

**目录**

[1 设计目标 4](#_Toc25689)

[2 系统分析 4](#_Toc23493)

[2.1 需求分析 4](#_Toc29899)

[2.2 功能分析 4](#_Toc2061)

[3 系统设计 4](#_Toc24250)

[3.1 组件设计 5](#_Toc26377)

[4 系统实现 5](#_Toc12301)

[4.1 功能 5](#_Toc23132)

[4.2 调试测试 8](#_Toc2504)

[5 项目分工 8](#_Toc29991)

[6 总结 9](#_Toc30735)

[6.1 关键特性 9](#_Toc19065)

[6.2 实现细节 1](#_Toc838)0

[6.3 结论 1](#_Toc17520)1

合肥学院软件工程专业

《程序与算法综合设计》任务书

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 项目名称 | 排序算法的实现与性能比较 | | | | |
| 学生姓名 | 钟健  徐杰东 | 学院 | 人工智能与大数据学院 | 专业班级 | 22软工2班 |
| 设计的主要内容、基本要求 | 【问题】  下图所示是一个城堡的地形图。编写一个程序，计算城堡中一共有多少个房间，最大的房间有多大。城堡被分割成个方块，每个方块可以有0~4面墙。“#”代表墙壁，“|”和“-”都表示没有墙壁。不被墙分割的方块连在一起组成一个房间。城堡外围一圈都是墙。    **【要求】**   1. 输入两个整数R和C，分别是南北向、东西向的方块数，在图形界面上自动生成上图的初始场景，默认都是“#”； 2. 设计人机交互模式，在图形界面中进行无墙壁的设置，无墙壁的地方将被设置为“|”或“-”；   计算城堡中的房间数，在图形界面上用黄色展示房间，用橙色展示最大的房间。 | | | | |
| 参考文献 |  | | | | |
| 备注 |  | | | | |

**城堡问题**

# 设计目标

编写一个程序，计算城堡中一共有多少个房间，最大的房间有多大。城堡被分割成个方块，每个方块可以有0~4面墙。“#”代表墙壁，“|”和“-”都表示没有墙壁。不被墙分割的方块连在一起组成一个房间。城堡外围一圈都是墙。

要求：

1. 输入两个整数R和C，分别是南北向、东西向的方块数，在图形界面上自动生成上图的初始场景，默认都是“#”；
2. 输入两个整数R和C，分别是南北向、东西向的方块数，在图形界面上自动生成上图的初始场景，默认都是“#”；
3. 计算城堡中的房间数，在图形界面上用黄色展示房间，用橙色展示最大的房间。

知识点：Java GUI编程，数据结构。

# 系统分析

* 1. 需求分析

1. 用户可以输入场景的大小，系统将生成初始场景。
2. 用户能够看到场景中的每面墙都可通过交互将其去除。
3. 用户可以随时点击按钮，计算城堡中的房间数，在图形界面上用黄色展示房间，用橙色展示最大的房间。
4. 用户可以看到更新后的场景与城堡中的房间数以及最大房间大小。
   1. 功能分析

用户界面：GUI允许用户输入场景大小，选择墙去除，并触发并查集过程。按钮交互后可显示更新后的场景与房间数和最大房间数。

并查集算法实现：实现了查找最顶上父亲方法与启发式合并方法。

# 系统设计

* 1. **组件设计**

**Castle：**这是主类，负责启动整个程序。其中定义了城堡的行数、列数、房间数量等属性，以及初始化方法、主窗口类 **MyJFrame** 的内部类，以及边类 **MyJButton** 和房间类 **MyJPanel**。

**MyJFrame类**：这是主窗口类，继承自 **JFrame**，用于显示城堡布局和提供交互功能。它包括一个按钮用于计算城堡信息，一个面板用于展示城堡布局，以及一些辅助方法用于计算窗口大小、添加房间、添加墙等。

**MyJButton：**这是表示城堡边的类，继承自 **JButton**，用于在城堡布局中显示边，并提供鼠标点击事件的监听器 **MyActionListener**，用于处理边的点击事件。

**MyJPanel：**这是表示城堡房间的类，继承自 **JPanel**，用于在城堡布局中显示房间，并包含房间的索引和标识属性。

**DSU类：**这是并查集类，用于处理城堡中房间的合并和查找操作，以及计算最大房间大小。

**控件和事件：**负责在图形界面中实现城堡布局的显示和交互功能。

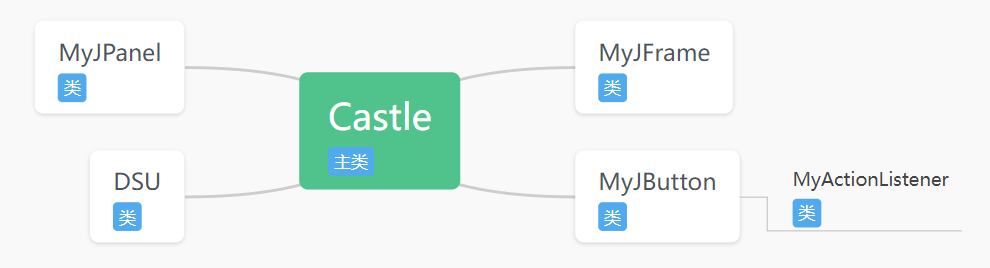


图 5-1 大致框架

# 系统实现

* 1. 功能

1. ****城堡布局初始化****：

用户可以输入行和列的数量来初始化城堡的布局。输入的行和列数量必须在1到50之间。初始化时，程序创建了一个MyJFrame对象，该对象表示城堡的主窗口，并在其中放置了房间面板和计算按钮。

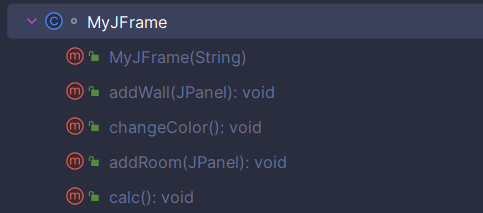


图 6-1 MyJFrame

1. ****城堡房间显示****：

程序通过 MyJPanel 类表示城堡中的房间，每个房间是一个矩形区域。房间使用 JPanel 控件创建，并在主窗口的面板中显示出来。



图 6-2 MyJPanel

1. ****城堡边界设置****：

程序通过 MyJButton 类表示城堡中的边界，即连接两个房间的边。边界使用 JButton 控件创建，并在主窗口的面板中显示出来。

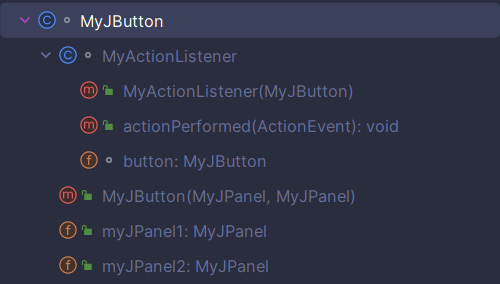


图 7-1 MyJButton

1. ****房间合并功能****：

用户点击两个房间之间的边界按钮时，程序会将这两个房间进行合并操作。合并操作由 DSU 类实现，使用并查集数据结构来管理房间的连接关系。

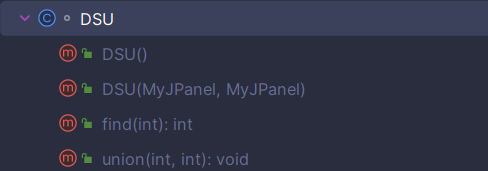


图 7-2 DSU

1. ****计算城堡属性****：

程序在合并房间时会实时计算城堡的房间数量和最大房间大小，并在点击计算按钮后弹出窗口显示。

1. ****界面交互****：

用户可以通过点击按钮进行房间的合并操作。用户点击计算按钮时，程序会计算并显示城堡的房间数量和最大房间大小。

通过这些功能，用户可以通过图形界面来直观地展示和交互城堡布局，以及了解城堡的属性信息。

* 1. 调试测试



图 8-1 运行结果显示

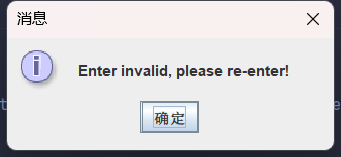


图 8-2 用户非合法输入，运行结果

# 项目分工

**钟健**：负责UI设计和开发，包括主窗口的布局和外观、按钮和面板的设计，以及用户交互事件的处理。具体任务包括：

1. 设计和实现主窗口的布局和外观，确保界面美观、直观和易用。
2. 设计和实现按钮和面板的布局、样式和交互效果，如颜色、大小、位置等。
3. 实现鼠标点击事件的监听和处理逻辑，包括按钮的点击事件和弹窗的显示。
4. 设计和实现用户输入的验证逻辑，如输入行和列数的验证，以确保输入的有效性。

**徐杰东**：负责业务逻辑和数据处理，包括房间合并逻辑、并查集算法的实现，以及房间状态的更新和展示。具体任务包括：

1. 实现房间合并的逻辑，包括判断两个房间是否相邻、合并房间、更新房间状态等。
2. 实现并查集数据结构和相关算法，用于处理房间的合并操作和集合查询。
3. 设计并实现房间状态的更新逻辑，包括颜色的变化、房间大小的更新等。
4. 实现计算最大房间大小和房间数量的功能，以便在合并房间时进行更新和展示。

# 总结

* 1. 关键特性

****用户友好的界面****：通过Swing库创建了一个简单直观的图形用户界面(GUI)，用户可以在界面上输入行数和列数，然后点击按钮进行房间合并。

****城堡房间布局****：根据用户输入的行数和列数，动态生成了对应数量的城堡房间，并在界面上展示出来，以便用户进行操作。

****并查集数据结构****：使用并查集数据结构管理房间的连接关系，确保相邻的房间可以合并。这样可以在合并房间时快速确定房间所属的连通分量，以及合并后的房间大小。

****房间合并模拟****：根据用户点击按钮事件，模拟房间的合并过程，并在界面上展示合并后的结果。通过改变房间的颜色来区分不同的房间状态，提高了可视化效果。

****输入验证****：对用户输入的行数和列数进行了合法性验证，确保输入在1到50之间，若输入不合法则会提示用户重新输入，保证了程序的健壮性和稳定性。

****界面布局优化****：通过动态计算房间面板的大小和位置，使得房间面板可以在窗口中居中显示，并根据窗口大小动态调整房间的大小和间距，提高了界面的美观性和灵活性。

****代码结构清晰****：采用了面向对象的设计思想，代码结构清晰，模块化程度高，易于理解和扩展，提高了代码的可读性和可维护性。

* 1. 实现细节

实现了一个名为Castle的程序，用于模拟城堡中的房间合并问题。程序的总体实现细节如下：

1. ****界面设计与交互****：

使用Swing库创建了一个GUI界面，包括一个主窗口和若干个房间面板。

主窗口使用了自定义的**MyJFrame**类，包含了一个**JPanel**作为容器，并设置了一个按钮用于触发计算和展示结果。

房间面板使用了自定义的**MyJPanel**类，每个房间面板表示一个房间，可以根据状态显示不同的颜色。

1. ****房间合并逻辑****：

使用并查集数据结构来管理房间的连接关系，确保相邻的房间可以合并。

每个房间面板（**MyJPanel**）具有一个唯一的ID，通过并查集中的**find**和**union**方法来判断房间之间的连接关系，并进行合并。

1. ****界面更新与交互反馈****：

当用户点击按钮时，通过按钮的监听器处理房间的合并逻辑，更新房间状态，并在界面上反馈结果。

根据合并后房间的大小和数量，更新界面上房间的颜色，以区分不同的房间状态。

通过弹窗反馈房间数量和最大房间大小等结果给用户。

1. ****输入验证和异常处理****：

程序会要求用户输入行数和列数，确保输入的范围在1到50之间，并对用户的输入进行验证，如果输入不合法，会弹出提示窗口要求重新输入。

1. ****其他细节****：

程序对界面的布局进行了优化，保证房间面板在窗口中居中显示，并且根据窗口大小动态调整房间的大小和间距。

通过合适的面向对象设计和代码封装，提高了代码的可维护性和扩展性。

通过以上实现细节，程序能够有效地模拟城堡中房间的合并问题，并提供直观的界面交互和结果展示，使用户能够清晰地了解房间合并的过程和结果。

* 1. 结论

这个Java项目实现了一个城堡房间合并模拟程序。用户可以在界面上输入指定行数和列数的城堡房间布局，然后点击计算按钮，程序会模拟房间合并过程，并给出合并后的结果。程序使用了并查集数据结构来管理房间的连接关系，确保相邻的房间可以合并。通过界面展示房间的合并过程，根据合并后房间的大小和数量，更新界面上房间的颜色以区分不同的房间状态。界面友好，用户可以通过点击按钮进行房间合并，并及时获知合并结果。对用户的输入进行了合法性验证，确保输入的行数和列数在1到50之间，若输入不合法则会提示用户重新输入。代码采用了面向对象的设计思想，通过自定义类和方法来实现功能模块化，提高了代码的可读性和可维护性。使用了Swing库来创建GUI界面，提供了良好的用户交互体验。界面布局经过优化，保证了房间面板在窗口中居中显示，并能根据窗口大小动态调整房间的大小和间距。代码结构清晰，易于扩展和修改，可以根据需要进一步添加新功能或优化现有功能。代码经过测试和调试，确保了功能的正确性和稳定性，能够满足用户的基本需求。