

# 项目说明文档

## 数据结构课程设计

——修理牧场

作者姓名：\_\_\_\_\_张翔\_\_\_\_\_

学        号：\_\_\_\_\_2352985\_\_\_\_\_

指导教师：\_\_\_\_\_张颖\_\_\_\_\_

学院、专业：\_\_\_\_\_计算机科学与技术学院 软件工程\_\_\_\_\_

同济大学

Tongji University

二〇二四年十二月九日

# 1 项目分析

## 1.1 项目背景分析

在现代农业生产中，农夫修复牧场栅栏是一项常见的任务。为了满足修复需求，农夫需要将一根长木头锯成若干块具有指定长度的小段。根据题目设定，锯木的酬金与每次锯木的长度成正比，即每次锯木的成本等于所锯木头的总长度。问题的关键在于如何优化锯木的分割策略，最小化总的酬金支出。具体而言，若木头的分割顺序不同，将导致不同的酬金支出。因此，如何通过合理的分割顺序，达到最小化费用的目标，是该问题的核心。这一问题涉及到动态规划、分治策略以及最优决策理论，具有较高的理论价值和实际应用意义。

## 1.2 项目需求分析

本项目的核心需求是通过优化木头分割的顺序，计算出将木头锯成  $N$  段的最小总酬金。每次锯木的费用由锯开的木头长度决定，而每次锯木的顺序与方案选择将直接影响后续操作的成本。系统需能够对不同的锯木方案进行评估，最终输出最小的总花费。

## 1.3 项目功能分析

项目具体功能分析如下：

- (1) 获取用户输入，确保输入数据的有效性，还需要对用户输入的数据范围进行校验。
- (2) 使用优先队列（最小堆）存储木头的长度，保证每次能快速获取当前最短的两段木头。
- (3) 基于贪心策略，不断从优先队列中取出最短的两段木头，将它们合并为一段，并计算当前的切割费用。
- (4) 合理输出总费用，清晰地显示最小切割成本，便于用户理解。

# 2 项目设计

## 2.1 数据结构设计

优先队列是一种特殊的队列数据结构，支持按照元素优先级进行排序和操作，通常采用二叉堆（最小堆或最大堆）作为底层数据结构。在修理牧场项目的背景下，优先队列具有独特的应用价值，能够高效地维护锯木问题中的最优分割点，从而减少总酬金。

## 2.2 MyPriorityQueue 类的设计

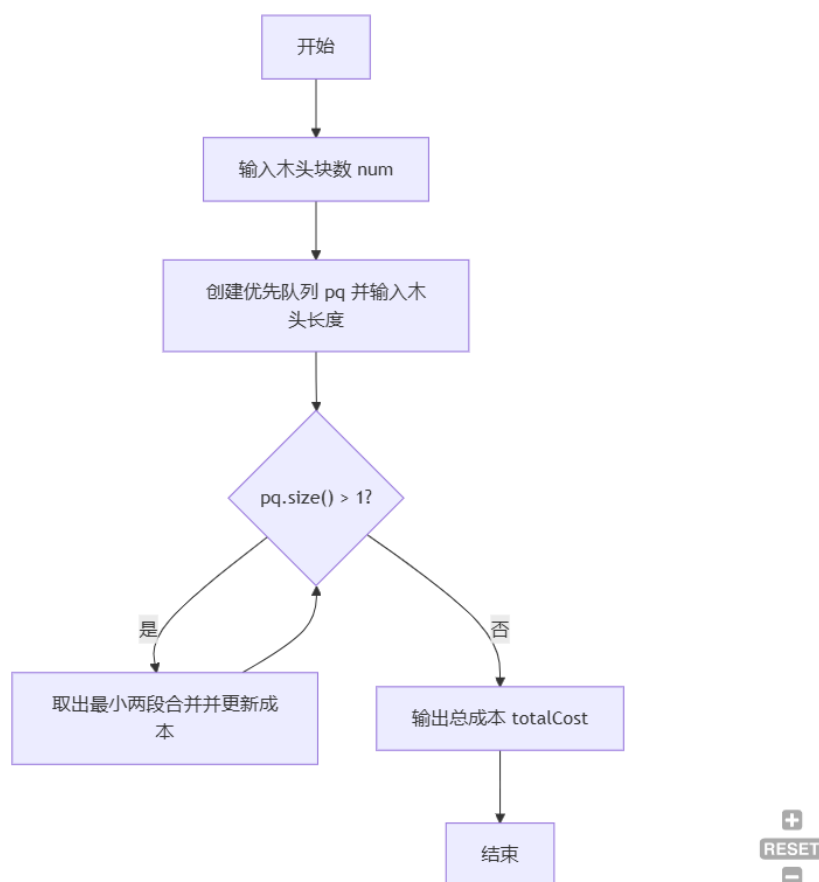
### 2.2.1 概述

MyPriorityQueue 是一个基于堆结构实现的模板类，用于表示一个通用的优先队列。该类通过动态数组管理元素，支持对任意类型的数据进行优先级排序，并提供插入、删除、获取队首元素等核心操作。类内部维护了堆化操作的私有成员函数 `heapifyUp` 和 `heapifyDown`，分别用于插入后向上调整和删除后向下调整，以确保堆的有序性。用户可通过公开的接口检查队列是否为空或已满，获取队列的当前大小，并执行插入或删除操作，同时提供对队首元素的访问接口。MyPriorityQueue 的设计具备较高的灵活性和通用性，适用于需要高效优先级排序的场景，如任务调度、路径搜索和资源管理等。

### 2.2.2 类定义

```
template<typename Type>
class MyPriorityQueue
{
private:
    Type* elements;
    int maxsize;
    int currsz;
    void heapifyUp(int index);
    void heapifyDown(int index);
public:
    MyPriorityQueue(int _maxsize);
    ~MyPriorityQueue() { delete[]elements; }
    bool isEmpty(void) const { return currsz == 0; }
    bool isFull(void) const { return currsz == maxsize; }
    int getSize(void) const { return currsz; }
    bool insert(const Type& item);
    bool remove(Type& item);
    bool getFront(Type& item) const;
};
```

## 2.3 项目主体架构设计



2.3.1 项目主体架构流程图

## 3 项目功能实现

### 3.1 项目主体架构实现

#### 3.1.1 项目主体架构实现思路

这段代码实现了最小成本切割木头的问题，采用的是经典的贪心算法，通过优先队列（最小堆）来优化操作顺序，从而确保每次合并操作的代价最小。具体实现思路如下：

程序首先读取用户输入的木头数量，并初始化一个优先队列用于存储木头的长度。接着，用户依次输入每段木头的长度，这些长度会被插入优先队列中。由于优先队列的特性，能够始终保证最小值优先出列。

在主循环中，程序每次从队列中取出最短的两段木头，将它们合并为一段新的木头，并将合并的代价累加到总成本中，同时将新木头的长度重新插入队列。这个过程不断重复，直到优先队列中只剩下一段木头为止。

最后，程序输出累计的最小切割成本。贪心策略在这里的核心是：每次优先合并最短的两段木头，这样可以保证局部最优，从而得到全局最优解。

### 3.1.2 项目主体架构核心代码

```
int main()
{
    int num = inputInteger(1, INT_MAX, "请输入要将木头切成的块数：");
    MyPriorityQueue<int> pq(num);
    int totalCost = 0;
    std::cout << std::endl << "请依次输入 " << num << " 段木头的长度：" ;
    for (int i = 0; i < num; i++) {
        int length;
        std::cin >> length;
        pq.insert(length);
    }
    while (pq.getSize() > 1) {
        int first, second;
        pq.remove(first);
        pq.remove(second);
        int cost = first + second;
        pq.insert(cost);
        totalCost += cost;
    }
    std::cout << std::endl << ">>> 最小切割成本为：" << totalCost << std::endl;
    return 0;
}
```

## 3.2 异常处理功能的实现

### 3.2.1 动态内存申请失败的异常处理

在进行动态内存申请时，程序使用 `new(std::nothrow)` 来尝试分配内存。`new(std::nothrow)` 在分配内存失败时不会引发异常，而是返回一个空指针（`nullptr`），代码检查指针是否为空指针，如果为空指针，意味着内存分配失败，这时程序将执行以下操作：

(1)向标准错误流 `std::cerr` 输出一条错误消息"Error: Memory allocation failed.";

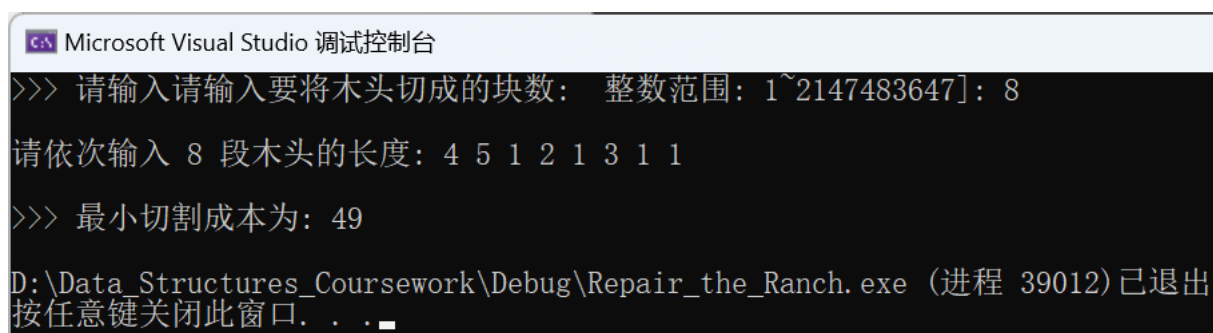
(2)调用 `exit` 函数，返回错误码-1，用于指示内存分配错误，并导致程序退出。

### 3.2.2 输入非法的异常处理

程序通过调用 `inputInteger` 函数输入木块的数量。`inputInteger` 函数用于获取用户输入的整数，同时限制输入必须在指定的范围内，函数的代码如下：

```
int inputInteger(int lowerLimit, int upperLimit, const char* prompt)
{
    std::cout << ">>> " << "请输入" << prompt << " 整数范围: [" << lowerLimit << "~"
<< upperLimit << "]: ";
    int input;
    while (true) {
        std::cin >> input;
        if (std::cin.good() && input >= lowerLimit && input <= upperLimit) {
            std::cin.clear();
            std::cin.ignore(INT_MAX, '\n');
            std::cout << std::endl;
            return input;
        }
        else {
            std::cerr << ">>> " << prompt << "输入不合法，请重新输入!" << std::endl;
            std::cin.clear();
            std::cin.ignore(INT_MAX, '\n');
        }
    }
}
```

## 4 项目测试

A screenshot of the Microsoft Visual Studio debug console. The title bar reads "Microsoft Visual Studio 调试控制台". The console output shows a program that prompts for the number of logs to cut, then for the lengths of 8 logs, and finally outputs the minimum cutting cost. The user input is 8 for the number of logs and 4 5 1 2 1 3 1 1 for the lengths. The program outputs 49 as the minimum cost. At the bottom, it shows the program path and that it has exited.

```
>>> 请输入请输入要将木头切成的块数: 整数范围: 1~2147483647]: 8
请依次输入 8 段木头的长度: 4 5 1 2 1 3 1 1
>>> 最小切割成本为: 49
D:\Data_Structures_Coursework\Debug\Repair_the_Ranch.exe (进程 39012) 已退出
按任意键关闭此窗口. . .
```

### 4.1 项目功能测试示例

## 5 集成开发环境与编译运行环境

Windows 系统: Windows 11 x64

Windows 集成开发环境: Microsoft Visual Studio 2022 (Release 模式)

Windows 编译运行环境: 本项目适用于 x86 架构和 x64 架构