**编程文档**

**(1) 该问题可以采用哪种解决问题策略解决？结合相关概念进行分析。**

**解决策略：动态规划结合单调栈**  
该问题需要找到矩阵中最大的全白矩形（所有元素为1）。分析如下：

1. **问题分解**：将二维问题转化为多个一维问题。对每一行，计算该行上方连续1的高度（形成直方图），然后在直方图中求最大矩形面积。
2. **动态规划**：维护高度数组 **height[j]** 表示第 **j** 列从当前行向上连续的1的个数（高度）。遍历每一行更新高度数组：
   * 若 **matrix[i][j] = 1**，则 **height[j] += 1**
   * 若 **matrix[i][j] = 0**，则 **height[j] = 0**
3. **单调栈优化**：对每行的高度数组，使用单调递增栈快速计算直方图最大矩形面积：
   * 栈存储索引，保持高度递增
   * 当遇到较小高度时，弹出栈顶并计算面积（高度 × 宽度）
4. **全局最优**：遍历所有行，更新全局最大面积。

**关键概念**：

* **重叠子问题**：每行的高度计算依赖上一行结果
* **最优子结构**：直方图最大矩形可通过局部计算得出
* **单调栈**：高效找到左右边界，时间复杂度 O(N)

**(2) C++ 代码实现与解释**

// 计算直方图最大矩形面积

int getMaxArea(vector<int> heights) {

    heights.push\_back(0); // 添加哨兵值

    stack<int> st;

    int maxArea = 0;

    for (int i = 0; i < heights.size(); i++) {

        while (!st.empty() && heights[st.top()] > heights[i]) {

            int h = heights[st.top()]; // 当前高度

            st.pop();

            int left = st.empty() ? -1 : st.top(); // 左边界

            int width = i - left - 1; // 宽度

            maxArea = max(maxArea, h \* width);

        }

        st.push(i);

    }

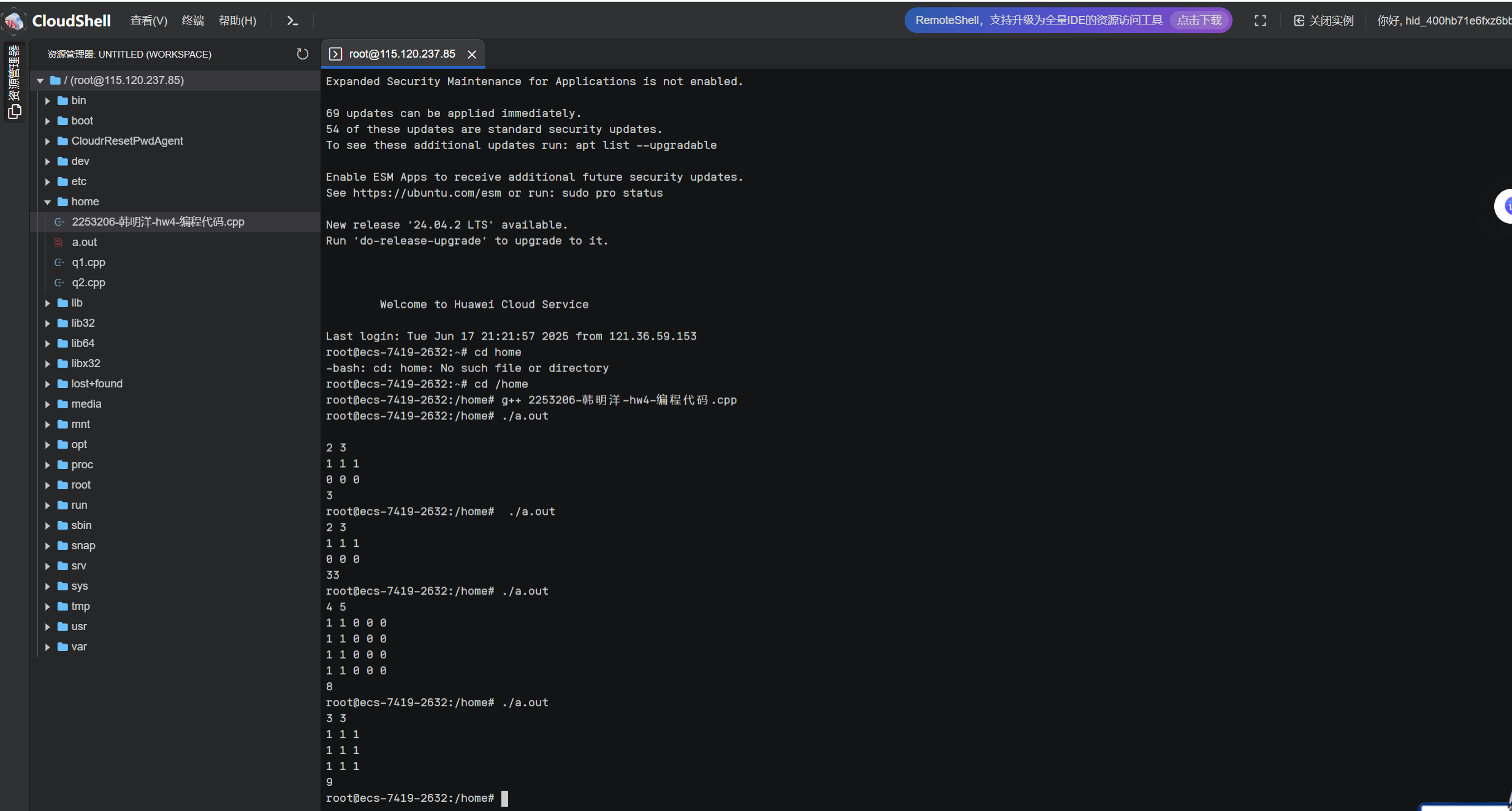
    return maxArea;

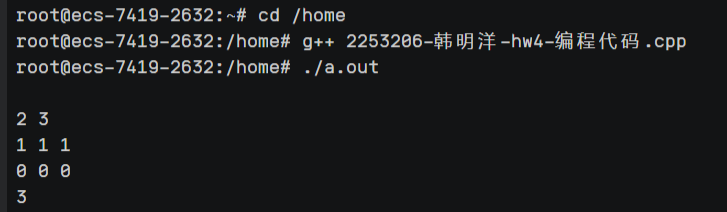
}

**代码解释**：

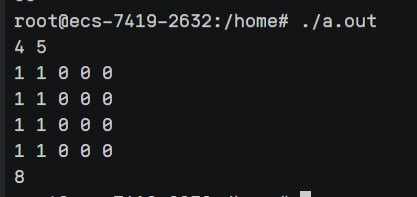
* **getMaxArea 函数**：
  + 输入：直方图高度数组 **heights**
  + 处理：添加哨兵值0确保栈清空，使用单调栈计算最大面积
  + 输出：直方图最大矩形面积
* **主函数**：
  1. 读入矩阵行数 **m** 和列数 **n**
  2. 初始化高度数组 **height**（存储每列连续1的高度）
  3. 遍历每一行：
     + 更新高度数组（遇1增加高度，遇0重置）
     + 调用 **getMaxArea** 计算当前行最大矩形面积
  4. 输出全局最大面积

**(3) 测试用例截图**

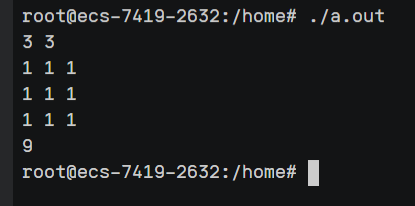
****



输出：**3**  
说明：第一行全1形成3×1矩形



输出：**8**  
说明：左侧4×2全白矩形



输出：**9**  
说明：整个3×3矩阵为全白矩形

**(4) 示例2的求解过程与中间表格**

**输入矩阵（修正后）**：

1 1 0 0 0

1 1 0 0 0

1 1 0 0 0

1 1 0 0 0

高度数组变化与最大面积计算：

行号 高度数组 直方图最大面积 计算说明

0 [1, 1, 0, 0, 0] 2 前两列高度1，宽度2（1×2）

1 [2, 2, 0, 0, 0] 4 前两列高度2，宽度2（2×2）

2 [3, 3, 0, 0, 0] 6 前两列高度3，宽度2（3×2）

3 [4, 4, 0, 0, 0] 8 前两列高度4，宽度2（4×2）

**(5) 时间复杂度分析**

1. **getMaxArea 函数**：
   * 遍历高度数组 O(N)
   * 每个元素入栈/出栈各一次 O(1) 均摊
   * **总复杂度**：O(N)
2. **主函数**：
   * 遍历所有行 O(M)
   * 每行更新高度数组 O(N)
   * 每行调用 **getMaxArea** O(N)
   * **总复杂度**：O(M × N)