并行计算 II 2024 春课程大作业
优化连通域标记算法

陶思睿　罗昊

1　连通域标记

连通域标记是图像处理和计算机视觉中的基础算法，用于将图像中的像素点分组成连通的区域。具体而言，如 Fig. 1所示，在本次作业中，如果两个点在彼此的八邻域内，我们就认为它们应属于同一类。针对给定的输入图片，该算法将给每个像素点打上标签，使得同一类的点标签相同，不同类的点标签不同（显然，我们不关心具体标签值的大小，只要满足以上要求即是正确的标记方法）。

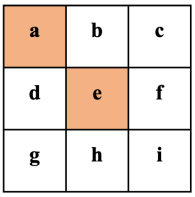

Fig. 1　点 a, e 连通；点 b, c, d, f, g, h, i 连通

常见的求解连通域标记的方法是使用并查集（Union-Find）。

1.1　并查集（Union-Find）

并查集是一种数据结构，用于处理一些不交集的合并及查询问题。在连通域标记问题中，它可以有效地判断两个小集合是否属于相同的大集合，并不断合并能够被合并的小集合以得到最终的连通域。

在使用并查集时，我们通常引入一个额外数组，用于记录每个元素的父节点。递归遍历元素的父节点，用得到的根节点下标指代该元素所属的集合。

1.1.1　并查集的基本操作

- **Init**：每个元素的父节点初始化为自己，即每个元素对应一个集合。
- **Find**：寻找某个元素所属的集合，如果该元素的父节点是自己，那么返回该元素；否则，递归地返回其父节点。
- **Union**：将两个元素所属的集合合并。

1.1.2　路径压缩优化

显然，并查集是一种树结构，这颗树的高度影响着 **Find** 和 **Union** 的效率。路径压缩是一种优化手段，它在执行 **Find** 操作时，递归地将所有经过的非根节点直接连接到根节点，这样能显著减少树的高度，使得后续操作均为常数复杂度。具体实现可以参考附件 serial.cpp。

1.2　求解算法描述

使用并查集求解连通域标记算法的核心步骤如下：

1. 初始化并查集，使每个像素点都是独立的集合。

2. 遍历图像中的每个像素点，分别检查其八邻域内的像素点（由于元素间的对称性，我们实际只需检查八个方向中的一半）。如果两个像素点具有相同的数值，则使用 **Union** 函数将它们合并。

3. 最后，对每个像素点执行一遍 **Find** 操作。通过路经压缩，确保同一集合内的像素点都直接指向根节点（代表其所属类别）。

由于路经压缩的特性，该算法能以 $\mathcal{O}(n^2)$ 的复杂度求解连通域标记问题。

2 作业要求

2.1 附件描述

作业附件内主要包含以下几项内容：

1. serial.cpp：串行算法示例。

2. pixel_n.txt：串行算法输入。n = 1, 2, 3, 4 分别对应四个处理好的图片输入，需要**分别**在四个输入上测试并行算法的性能。_debug 仅作为调试辅助。

3. visual.py：用于可视化连通域标记结果。该方法首先赋予每个连通域**唯一确定**的类别值（最小下标值，因此每幅图片最终输出的图像颜色也是**唯一确定**的），然后哈希映射到 RGB 色彩空间输出图像。**【不要求**运行和优化，仅提供作为展示，效果图见 Fig. 2**】**。

4. convert_image.py：助教用于图像二值化和导出的脚本。大家闲了可以自己玩玩。

5. 其他用于编译、提交集群作业等的辅助文件。

2.2 作业要求

作业要求自选使用 MPI / OpenMP / MPI+OpenMP 或 CUDA 实现并行版本的连通域标记算法，并提供优化报告。

- 并行代码需在四个算例上分别运行得到性能数据，将计算总用时作为排名。
- 基于并查集的算法实现仅作为参考，允许使用其他算法。
- 可改变读入方式。除文件 I/O、内存分配之外的部分均需记入耗时。
- 对于 MPI / OpenMP / MPI+OpenMP 的实现：基于数院集群，最多使用 4 个节点，总计最多使用 16 个核心。
- 对于 CUDA 的实现：基于数院集群，最多使用一张 GPU 卡。
- 性能分由排序记分得到，是成绩的最主要组成部分，请尽可能地优化你的程序。使用非数院集群，成绩视情况打折扣。
- 不限制编程语言、编译选项和编译器版本（推荐 C/C++ 实现）。
- 提交完整的并行程序代码，包含计时模块（多次运行取平均）/正确性验证，提供与非优化版本的性能加速比较和分析。
- 提交报告中要求说明硬件型号、编译器版本、划分方法以及每个线程/进程的任务负载。

- 鼓励报告中提供分阶段优化或者多版本优化的分析和比较。
- **资源紧张，请同学们合理规划作业时间！**

3 提交要求

将代码和报告打包后发送至邮箱：parco2024@163.com，邮件主题为“学号-姓名-第三次作业”，截止日期为 6 月 20 日 23:59。

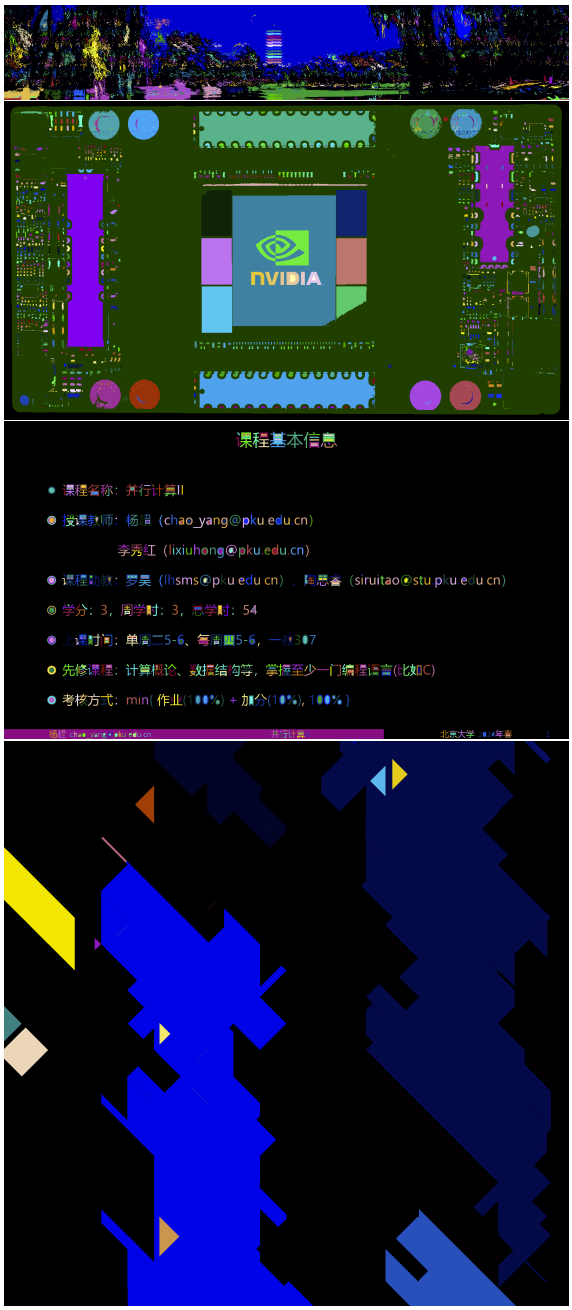

Fig. 2 四张图片输出可视化展示