Lab 1 实验报告

2021 年 11 月 26 日

1 问题描述

在上一个实验中，我们通过动态规划打表的方式完成了 LCS 的长度计算和结果输出，打表顺序是很自然的从左到右、从上到下，但这种顺序构建的循环很显然是有数据依赖性的，根据状态转移方程，每一个 $f(i, j+1)$ 都依赖于 $f(i, j)$ 的结果，因此我们在得到 $f(i, j)$ 之前无法提前开始计算之后的格子，换言之这种朴素的算法并不具有并行性。

于是在这次实验中，我们将根据以下状态转移方程，设计新的打表顺序打破现有数据依赖性，以得到可以并行的程序。

$$f(i,j) = \begin{cases} 0, & i < 1 \text{ or } j < 1 \\ f(i-1, j-1) + 1, & A[i] = B[j] \\ \max(f(i, j-1), f(i-1, j)), & A[i] \neq B[j] \end{cases}$$

2 算法设计

如下图所示，注意到每一个格子的值仅与其左、上、左上的格子有关，于是沿着反对角线的方向构造内层循环，每次内层循环计算一条反对角线，图中橙色表示已计算的格子，绿色表示正在计算的格子，灰色表示还未访问的格子，很明显绿色格子是可以并行计算的。外层循环沿对角线方向进行。

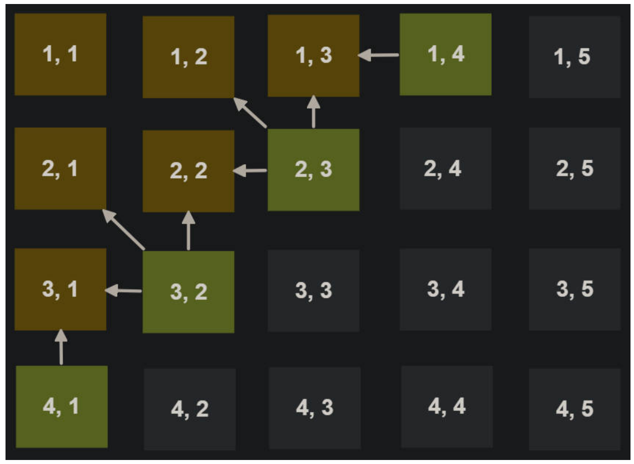

图 1: 反对角线

于是将二维表旋转至反对角线水平方向，假设原索引 (i, j) 分别代表行序与列序，则新构建的双层循环变量 (s, t) 满足 $s = i + j$ 和 $t = j - i$ 关系，具体实现只需再加一个偏移即可，再根据边界条件的区别分为三类，如图所示划分。

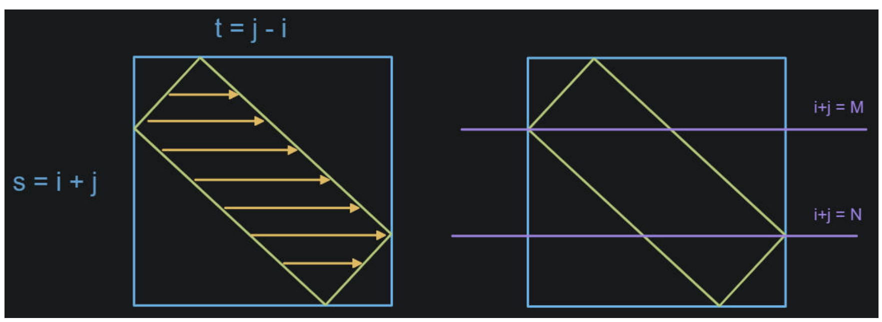

图 2: 索引关系

3 算法细节

- 受到 Lab0 参考答案的启发，设计使用一维数组存储动态规划表，空间复杂度减少至 $O(M+N)$。
- 一开始将 for 循环分三类写，代码长而丑，后经助教指导合并为一个 for 循环，在外层循环计算内层循环边界条件即可，美化代码，更具有可读性。
- 用三目运算符代替一些 if-else 分支，减少分支预测开销，且在 O3 优化中速度更快，实验表明有效。
- 在 cilk_for 前编译指示颗粒度大小为 2048 (经粗略调参所得)，减少并行产生的 block 数量，减少通讯开销。

4 实验结果

最早运行样例代码，在小阶数 (约 10^4 量级) 情况下出现反对角线并行速度慢于串行速度的异常情况。

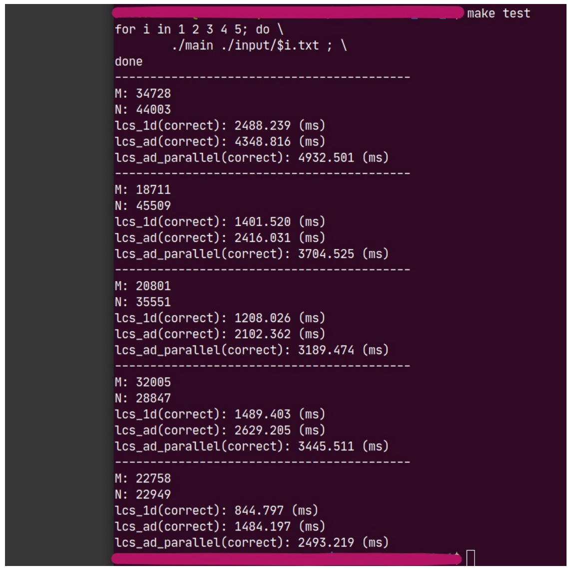

图 3: 小阶数测试结果

后将阶数增大 (约 10^5 量级) ，反对角线并行速度远快于串行，并且超过朴素算法速度，证明了并行计算在大阶数情况下的有效性。

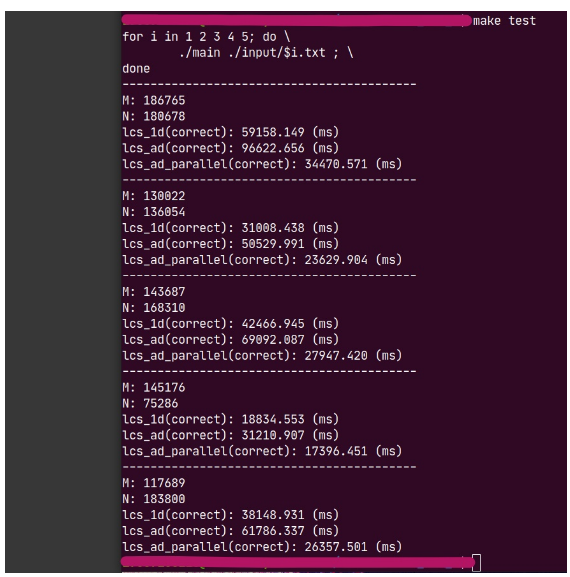

图 4: 大阶数测试结果

5 实验结果分析

- 小阶数情况下反对角线串行速度慢于朴素串行算法，可能是因为反对角线空间不连续导致的 cash miss 代价高于 O3 优化带来的加速，反对角线并行慢于反对角线串行的原因，可能是并行通讯开销代价高，总体不如 pipeline 优化速度。
- 大阶数情况下反对角线并行时间复杂度 $O(n \log n)$，反对角线串行时间复杂度 $O(n^2)$，在 n 足够大时，会克服常数项趋向该加速比。