Lab 2：并行化 LCS

最长公共子序列问题（Longest Common Subsequence Problem）

LCS 是非常经典的动态规划算法问题，这裡简单介绍一下。假设给定输入两个字符串 S1 和 S2 ，要求输出两者的最长公共子序列。从数学上来说，一个序列 $A_1, A_2, ..., A_n$ 的子序列就是按照顺序挑选出部分元素 $A_{n1}, A_{n2}, ..., A_{nk}$ 所构成的序列。因此，最长公共子序列，就是指一个序列 S' 都是两个输入序列的子序列，并且是所有可能的子序列里面最长的子序列。

不熟悉算法设计的同学不用紧张，这次实验不涉及如何求解 LCS ，只要求同学并行求解以下 LCS 求解公式：

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

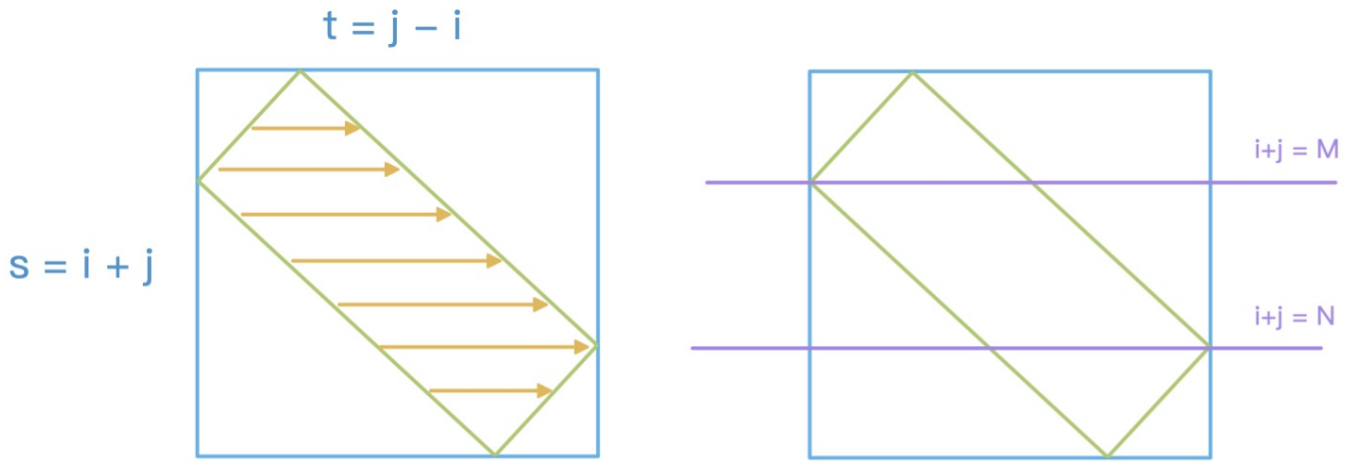
f [i, j] 表示 A[1...i] 和 B[1...j] 的 LCS 长度。朴素的求解 LCS 方法是：建立一张二维表 f，从左到右（f [*, 1] 到 f [*, n]）、从上到下（f [1,*] 到 f [n,*]）根据递归公式逐行求解 f [i, j]。但是仔细观察数据依赖相关性后发现，每一格 f [i, j] 都依赖于左边格子 f [i, j–1] 的计算结果，然而我们却是从左到右依序计算，计算完 f [i, j] 前不能提前计算 f [i, j+1] 与其他格子。因此，这种朴素 LCS 算法不能充分利用处理器的多核性能。

任务内容

1. 运用课堂上各式各样并行优化的技巧，打破计算 LCS 长度的数据依赖性，并且使用 Cilk 框架（或是 OpenMP、pthread 等）编写求解 LCS 长度的程序（完成框架代码 lcs.cpp 中三个函数），尽可能充分利用处理器的性能。
2. 撰写实验报告，说明你所使用的优化技巧、分析新算法的加速比、实验对比优化前后的性能变化。

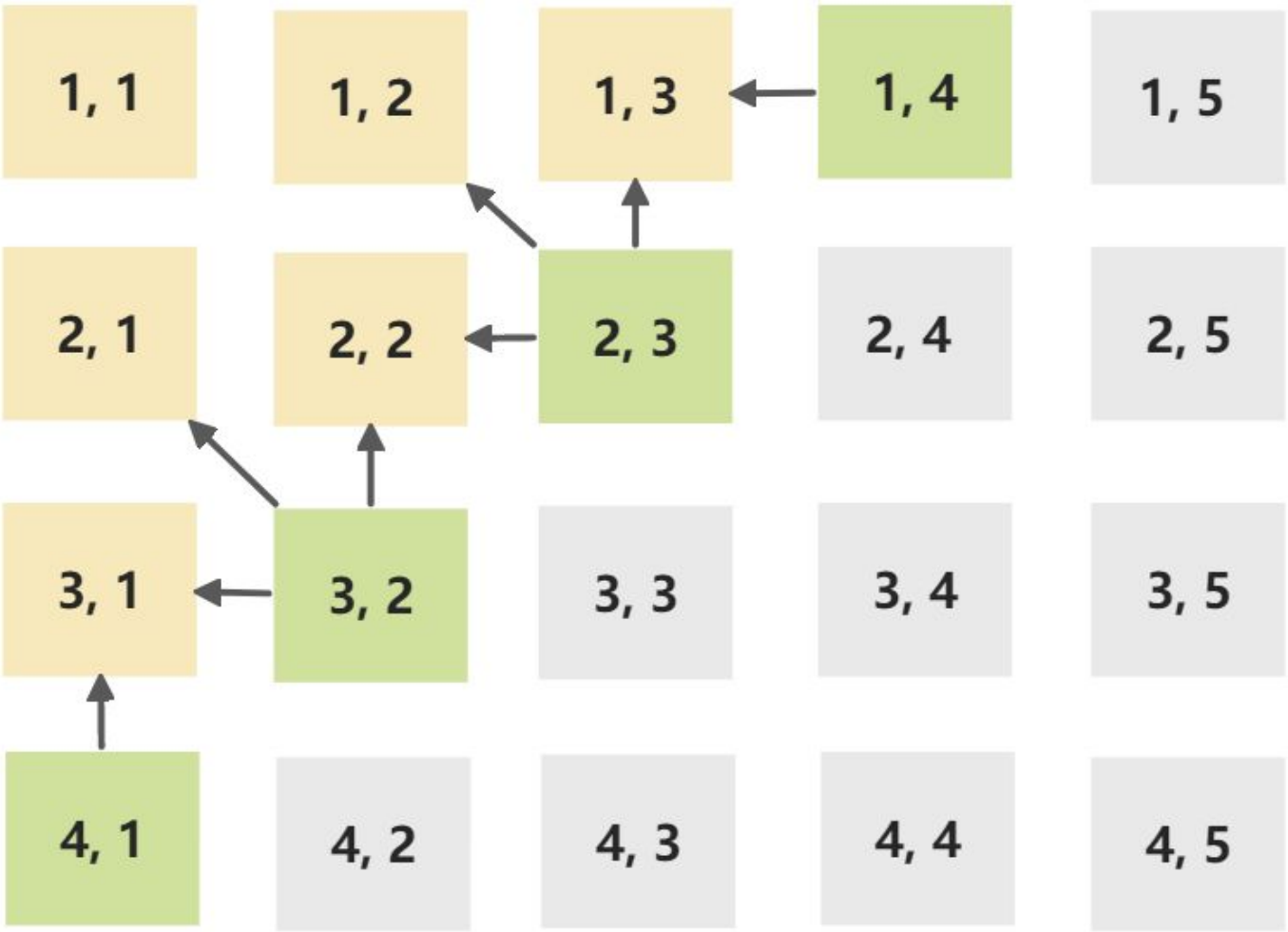
提示：LCS 反斜对角线方法

打破计算 LCS 的数据相关性的方法很多， R. A. Chowdhury and V. Ramachandran, Cache–oblivious dynamic programming, 2006 就有提出一种方法，有兴趣的各位同学可以参考一下。
这裡提示一种相对简单的 LCS 算法优化思路。观察一下 f(i, j) 的数据相关性，右下方所有格子不能提前计算，左上方所有格子必定已经计算得出，能够与 f(i, j) 同时计算的格子必定位于 f(i, j) 左下方与右上方。这个简单的事实让我们有了简单的数据并行计算的方法：LCS 反斜对角线方法。



LCS 反斜对角线方法就是按照反斜对角线的方向逐条对角线计算，因为反斜对角线上每个元素彼此之间都没有数据依赖关系，可以同时计算整条反对角线上所有元素。下面简单画了一张图展示 LCS 反斜对角方法，灰色表示尚未计算的格子，黄色表示已经计算完成的格子，绿色表示当前反斜对角 i + j = 5 的格子。

但是这种方法存在着什么样的问题呢？同学们可以计算这种方法的加速比与通信复杂度，想想看有没有进一步改进的方法。



辅助框架

这次实验提供了辅助的框架代码，请在群里下载实验框架。实验框架可以自动测试正确性和性能，帮助各位同学完成实验。由于框架代码是助教手写，儘管做过一些测试去验证框架的正确性，但可能仍然存在未知的错误，因此遇到任何问题请即时反馈。

实验框架使用了 make 自动编译工具，自动生成输入测试数据并且将文件夹底下所有代码文件编译成一个可执行文件。使用 Linux/MacOS 同学可以简单透过 apt/brew 等软件简单安装 make 工具。Windows 系统下也可以使用 make 工具，相关讨论 How to install and use "make" in Windows? 供各位参考。

简单介绍一下我们提供的三个 make 指令和说明相关使用注意事项：

1. make all (all可以缺省) 指令将文件夹底下所有 .cpp 文件编译为可执行文件 main。
2. make random 自动随机生成多个测试文件。测试文件个数 TEST_NUM 和最大输入长度 INPUT_MAX_LEN 可以在 Makefile 里进行修改。
3. make test 按照测试文件的编号顺序，依次对输入进行性能与正确性测试。
4. 如果想要使用 Cilk 工具 Cilksan 和 Cilksacle，请在 Make 将 CFLAGS 加上 –fsanitize=cilk –fcilktool=cilkscale 编译选项，并且在代码中引入 cilk/cilkscale.h 和 cilk/cilksan.h 头文件。
5. MacOS 系统上的同学请记得将 Makefile 的 CC 编译器选项修改为 xcrun [your–OpenCilk–clang++]。

如果无论如何都搞不定 make 工具，这也没什么关系，因为我们代码文件数量本身就不多，直接编译所有代码的命令行指令也很简单。总之，挑选一种适合自己的代码编写方式，顺利完成实验就行了。

实验提交事项

请在 **11/27 23:59 前**提交代码和实验报告至服务器，**逾时不候**。请完成框架代码 lcs.cpp 中 lcs_basic、lcs_ad、lcs_ad_parallel 三个函数，他们分别对应基本算法、反斜对角线串行算法、反斜对角线并行算法。每一个函数都有两个输入字符串和输入长度，函数要求返回正确的 LCS 长度。基本算法可以参考 Lab 0 解答的助教的代码。

注意！在 Docker 或 WSL 等**虚拟机**中运行代码，任何的并行相关性能测试都是**不准确的**！请到实机上进行性能测试，虚拟机上只能检验代码正确性。

祝各位好运 ：)