Motivating Application: Parallel Linear Algebra 激励应用：并行线性代数

While MANY math topics show up in computer science, perhaps no area of math is more important for modern trends in computer science than Linear Algebra. From computational geometry, to graphics and animation, to robotics, to machine learning, to data analysis, Linear algebra is increasingly useful. As a result, it's helpful to accelerate basic linear algebra operations in any ways we can. For the exercise, we will use what we've seen about parallelizing map and filter to implement parallel versions of dot product and matrix multiplication methods. Don't worry, though, no knowledge of linear algebra is necessary for this exercise! We'll describe the operations you're meant to implement!虽然计算机科学中出现了许多数学主题，但对于计算机科学的现代趋势来说，也许没有哪个数学领域比线性代数更重要。从计算几何到图形和动画，再到机器人技术，再到机器学习，再到数据分析，线性代数越来越有用。因此，以任何可能的方式加速基本的线性代数运算都是有帮助的。对于这个练习，我们将使用我们所看到的关于并行化映射和过滤器的内容来实现点积和矩阵乘法方法的并行版本。不过，别担心，这个练习不需要线性代数知识！我们将描述您要实施的操作！

In this assignment we're also going to implement a filter operation on an array of Strings, which... doesn't match that theme. There are limits to my creativity! 😳在这个任务中，我们还将对字符串数组实现一个过滤操作，这...与该主题不匹配。我的创造力是有限的！

Problem Statements 问题陈述

You will be implementing three methods for this assignment. For each of them, we have provided an equivalent sequential implementation. The behavior of your methods should exactly match the behavior of these sequential versions, but should be in parallel! 您将为此作业实施三种方法。对于它们中的每一个，我们都提供了一个等效的顺序实现。你的方法的行为应该与这些顺序版本的行为完全匹配，但应该是并行的！

You will notice, though, that each method you are instructed to implement has one additional parameter beyond what its sequential version has – an input to define the sequential cutoff. Your implementations should be defined to work for any sequential cutoff given, so long as it is at least 1. 不过，您会注意到，除了顺序版本之外，您被指示实现的每个方法都有一个额外的参数——一个用于定义顺序截止的输入。你的实现应该被定义为适用于任何给定的顺序截断，只要它至少为1。

For DotProduct and MatrixMultiply we have suggested a class structure for using ForkJoin. For those you will need to decide what fields you will need for the compute methods, and will need to update the constructors accordingly. 对于DotProduct和MatrixMultiply，我们建议使用ForkJoin的类结构。对于这些，您需要决定计算方法需要哪些字段，并相应地更新构造函数。

For FilterEmpty we have not provided a class structure, so that will be up to you. We have, though, provided a parallel implementation of prefix sum that matches the algorithm presented in lecture. We suggest you use that implementation. 对于FilterEmpty，我们没有提供类结构，所以这将由您决定。不过，我们提供了一个与讲座中介绍的算法相匹配的前缀求和的并行实现。我们建议您使用该实现。

Without further ado, let's discuss the expected behavior of each method. 闲话少说，让我们讨论一下每种方法的预期行为。

1. `dotProduct`: The dot product of two vectors (arrays) is defined to be the sum of the products of their corresponding indices. Less succinctly, we calculate the dot product of arrays a and b by summing together a[i]*b[i] for every index i. For example, the dot product of [1, 2, 3] and [4, 5, 6] is

1*4+2*5+3*6=32. 两个向量（数组）的点积被定义为它们相应索引的乘积之和。更简洁地说，我们通过将每个索引i的a[i]*b[i]相加来计算数组a和b的点积。例如，[1,2,3]和[4,5,6]的点积为1*4+2*5+3*6=32。

2. `multiply`: Given two square matrices (n-by-n arrays) their product will be a matrix of the same size (so also an n-by-n array). The value at index i,j of the product of matrices a and b is defined to be the dot product of row i of a and column j of b.给定两个方阵（n-by-n阵列），它们的乘积将是一个大小相同的矩阵（n-byn阵列也是如此）。矩阵a和b的乘积在索引i、j处的值被定义为a的第i行和b的第j列的点积。
3. `filterEmpty`: Given an array of Strings, this method should return a new array of strings containing only the non-empty strings from the original. This relative order of the remaining strings should remain the same. The length of the output array should match the number of non-empty strings in the input.给定一个字符串数组，此方法应返回一个新的字符串数组，其中仅包含原始字符串中的非空字符串。其余字符串的相对顺序应保持不变。输出数组的长度应与输入中非空字符串的数量相匹配。

Implementation Guidelines 实施指南

Your implementations in this assignment must follow these guidelines

You may not add any import statements beyond those that are already present (except for where expressly permitted in this spec). This course is about learning the mechanics of various data structures so that we know how to use them effectively, choose among them, and modify them as needed. Java has built-in implementations of many data structures that we will discuss, in general you should not use them.

Do not have any package declarations in the code that you submit. The Gradescope autograder may not be able to run your code if it does.

Remove all print statements from your code before submitting. These could interfere with the Gradescope autograder (mostly because printing consumes a lot of computing time).

Your code will be evaluated by the gradescope autograder to assess correctness. It will be evaluated by a human to verify running time. Please write your code to be readable. This means adding comments to your methods and removing unused code (including “commented out” code).

Provided Code

Several java classes have been provided for you in this zip file. Here is a description of each.

`Sequential` This class contains sequential versions of each method above. The behavior of the methods you write should match these, but should be parallelized using Java’s ForkJoin framework.此类包含上述每个方法的顺序版本。您编写的方法的行为应该与这些相匹配，但应该使用Java的ForkJoin框架进行并行化。
Do not submit this file

`TestClient` This class contains a main method. This main method invokes subroutines which compare your code’s behavior with the sequential code’s behavior by running both on arrays with random contents. We recommend you try varying the sizes of the arrays and selections of sequential cutoffs to further test your code.这个类包含一个main方法。此主方法调用子程序，通过在具有随机内容的数组上运行这两个子程序，将代码的行为与顺序代码的行为进行比较。我们建议您尝试改变数组的大小和顺序截断的选择，以进一步测试您的代码。
Do not submit this file

`ParallelPrefix` In this class we provide an implementation of the prefix sum algorithm that we presented in class. We recommend you use this when you do your filtering.在这个类中，我们提供了我们在类中提出的前缀

求和算法的实现。我们建议您在进行过滤时使用此功能。
Do not submit this file

`DotProduct` Implement the dotProduct method in this class using ForkJoin. You will submit this file to Gradescope

`MatrixMultiply` Implement the multiply method in this class using ForkJoin. You will submit this file to Gradescope

`FilterEmpty` Implement the filterEmpty method in this class using ForkJoin. You will submit this file to Gradescope

Part 1: dotProduct This method is the most straightforward of the three you're writing for this assignment, and in fact you will be adapting this code to help you out in part 2. So I recommend you definitely start here!

The input-output behavior of this method was described in the Problem Statements section above. If you find that description insufficient, however, it may be helpful to reference the sequential implementation in the Sequential.java class. Your goal is to use ForkJoin to write a parallel algorithm which is equivalent to the sequential.

Notice that dotProduct is an example of a reduce operation - the input is a pair of arrays, and the goal is to calculate a singular value by accumulating values as indices of those arrays. For this reason, to get started, I recommend referencing the various implementations of reduce operations you've seen so far. For example, the array summation method presented in class and the examples you worked on in section.

Make sure that you implement your method so that it will work properly for any choice of sequential cutoff! Part 2: Matrix Multiplication For this part things are going to get a bit trickier. We're actually going to use 2 different classes to make this super parallel! We will have one class that extends RecursiveTask and another that extends RecursiveAction.

The class that extends RecursiveTask will be a modification of the dot product method you completed for part 1. Instead of giving it two 1-dimensional arrays for the dot product, in this method we will give two square 2-dimensional arrays, a row, and a column. The method will then find the dot product of the given row of the first array with the given column of the second array. The private dotProduct method given in Sequential matches its expected behavior (and the sequential multiply method uses it in the same way we are about to use it). In the next step we will be parallelizing the computation of each cell of the product matrix, this method will be used to find each value in a parallel way.

The class that extends RecursiveAction will populate a given empty array with the product of two matrices. In order to achieve the desired span, we will need to be careful in how we divide our subproblems here. Each instance of this object will be responsible for some region of the 2-d array. Because there are now 2 dimensions to deal with we will need 4 values to identify which region of the matrices each RecursiveAction is responsible for. Whereas before we only needed a lower bound and upper bound index to identify our region, we will have bounds for the left, right, top, and bottom for our 2-d array. When dividing into subproblems, you should create 4 subproblems, each responsible for one quadrant of the array.

The same sequential cutoff as the one given in the multiply method should be used as the sequential cutoff for all compute methods. Part 3: Filter Empty The last method to implement is a filter/pack method. Our goal is to filter out all of the empty strings from a given array of strings. At a high level, here is how we can adapt the procedure we discussed in class to this problem: For the given array of strings, create a new array of ints

of the same length. Do a map operation on the array of strings to populate the array of ints such that empty strings map to 0 and non-empty strings map to 1. Perform a prefix sum operation to the match result Create an array of Strings whose length matches the last value in the prefix sum result Use all three of the original input array, the map array, and prefix sum array to populate this new array with only the non-empty strings Return that array We have provided a parallel implementation of prefix sum for you to use. I strongly encourage you to have a look at it so you can become familiar with the details of how it works! You will be responsible for designing and implementing all other parts of this algorithm.