

**课 程 项 目 报 告**

**项目名称： akari问题求解**

**专业班级： CS1605**

**学 号： U201614641**

**姓 名： 洪锦涛**

**指导教师： 陆枫**

**报告日期： 2019年7月29日**

**计算机科学与技术学院**

目录

[1 串行算法复杂度分析 1](#_Toc15335831)

[1.1 时间/空间算法复杂度分析 1](#_Toc15335832)

[1.2 算法描述 1](#_Toc15335833)

[2 分析并行实现的正确性 3](#_Toc15335834)

[2.1 可行算法的描述 3](#_Toc15335835)

[2.2 工作分解 3](#_Toc15335836)

[2.3 编程模型 3](#_Toc15335837)

[2.4 性能检查及优化 3](#_Toc15335838)

[3 大数据场景分析 4](#_Toc15335839)

[3.1 任务描述 4](#_Toc15335840)

[3.2 解答 4](#_Toc15335841)

[4 并行优化方案设计 5](#_Toc15335842)

[4.1 任务描述 5](#_Toc15335843)

[4.2 解答 5](#_Toc15335844)

# 1 串行算法复杂度分析

时间/空间算法复杂度分析

程序采用的矩阵均为二位数组，故空间复杂度为O(n^2)。

对于M个黑色有数字点，每个灯的数字不同最多能有6种不同的组合，此阶段最多需要进行6的M次方选择。之后剩下的白色块有N-M个，则最多需要进行2的N-M次方次选择，所以最终的时间复杂度最差为O（6n）。但由于使用了回溯法，运行过程中会不断剪枝所以时间复杂度会远小于这个。

算法描述

首先编写solveAkari函数，用来接收输入的矩阵，将所有的黑块放入一个vector方便后面进行黑块的递归，每一个黑块由一个struct来表示，struct里面包含这个黑块的坐标和值，然后对黑块进行冒泡排序进入黑块递归。

黑块递归的深度为存储黑块的vector的长度，按照下标进行递归，首先函数开始对每一个进入函数的矩阵进行判断，判断是否为错误的分支或者是否黑块已经全部都安排完毕，若为错误的分支则返回空集，若黑块已经全部安排，则对矩阵进行判断是否为正确的答案，若为正确答案则返回正确答案，否则进入白块递归；均不是则是按照黑块的值分情况，在每个情况下，再根据上下左右再分情况进行相应的放灯和点亮白块等操作，在根据后续递归的返回值来判断是继续递归还是返回正确或错误。

之后是白块递归，也是按照下标进行递归，白块递归是安排所有黑块之后，还有没有照亮的白块，此时也是像黑块那样，先使用一个vector存储所以的白块，每一个白块由一个struct来表示，struct里面包含这个白块的坐标然后进行递归，每次函数也会在开始对每一个进入函数的矩阵进行判断，若为错误的分支则返回空集；若白块已经全部安排，则对矩阵进行判断是否为正确的答案，若为正确答案则返回正确答案，否则返回空集；均不是则进行相应的放灯和点亮白块等操作，在根据后续递归的返回值来判断是继续递归还是返回正确或错误。

本次实验使用了五个辅助函数，第一个是light\_num，该函数用于获取周围灯的数量；第二个是light\_can\_num，该函数用于获取周围可插灯的数量；前两个函数主要是为了便于check和cut函数的判断。第三个是light\_white，即将穿入进来的灯的坐标周围未被挡住的白块点亮；第四个是check，该函数用于确定result是否为最终的答案；第五个是cut，该函数用于确定矩阵是否为错误的分支。

# 分析并行实现的正确性

可行算法的描述

对于每个不同的选择解与其它选择中的解必然不同，所以在后面的分析和继续求解上都是不同的。而不同选择之间并不需要数据的交换，每个选择的数据都源自于自己的父亲解结点，所以可以并行。

工作分解

对于每个可能进行选的结点，若其可以进行选择则进行选择创建多个线程继续调用不同的求解函数，如果问题最终无法求得解则返回，在串行程序中这样的方法可以成功求得解，则在并行的程序中，必然会有最终一个选择进入最终解，通过层层返回，最终得到正确解。为了节省线程创建开销，仅在黑块数字为三的时候进行并行操作。

编程模型

利用thread类来完成线程的创造和并行操作。

性能检查及优化



图2.1 串行运行结果

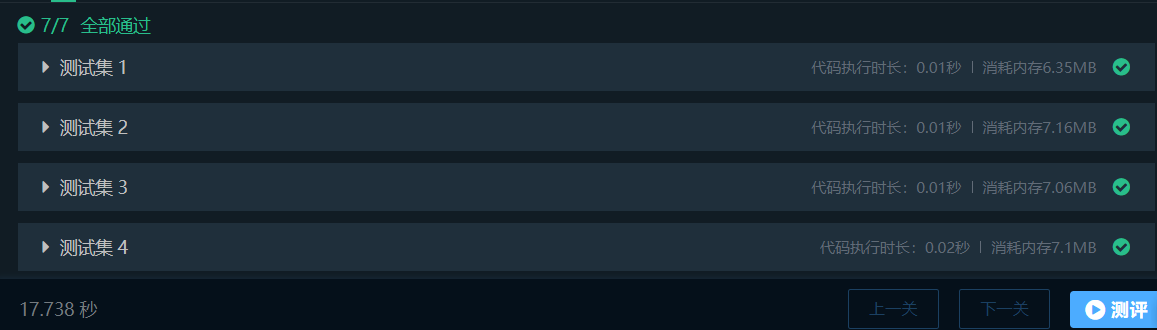


图2.2 并行运行结果

由于数据集过小，由于额外创建线程开销使得并行时间大于串行时间。

# 大数据场景分析

任务描述

1. 设计大数计算场景下”akari回溯”的测试用例并分别使用串行和并行的方式实现。

2. 分析并行实现的加速比。

3. 分析并行实现加速比的正确性

解答

当矩阵规模大到一定程度，矩阵的解大概率具有多种不同的解，利用多线程同步进行求解，最短时间为最先遇到正确的解的选择所需要的时间，而串行算法时间则为选择的分支为固定的顺序，求解的解时间不一定会是不同选择排列组合下最快能够求解出解，故并行的速度会快于并行速度的解。

# 并行优化方案设计

任务描述

1. 设计大数场景并行实现的优化方案。

2. 测试并分析优化方案的加速比与正确性。

解答

对并行粒度进行划分，设置为只在遇到的第一个黑块数字三进行并行操作，可以不再建立多余的线程求解而是以一种类似于串行程序的执行方式来继续求解可以有效减少线程建立的开销从而提高效率，太多的线程建立很明显拖慢了程序的速度。