CSE 326 Analysis and Design of Algorithms

Dr.Walid Gomaa

Name	**ID**
Mohamed Abdelmonem Makram	120220055
Abdelrahman Ahmed Shaheen	120220228
Abdelrhman Mohamed Eldenary	120220253
Anas Ihab Badr	120220360

Computer Science Engineering Department
Egypt-Japan University of Science and Technology

Contents

Schur Numbers Week 7 Report

March 28, 2025

1 Recap

In Week 5 we began to think about the idea of nested parallelism where we said we would use CUDA to try to brute force smartly enough where each thread would call other threads to complete the problem.

In Week 6 we began to learn CUDA and its basics. We compiled a simple program that uses our two pointer algorithm to check for the available colorings of the next number, which would be the first step in the nested parallelism approach. As each nested thread will then have a version of the problem with the next number colored differently.

Currently, in Week 7 before implementing the nested parallelism approach, we would try to add another layer of randomness to the algorithm which we believe may help in getting results. We will explain this in the next section.

2 Randomness

Since our main goal is to try to improve the lower bound for $S(6)$, we would be working on the already existing coloring of $S(5)$. But as can be seen in the comparison between $S(4)$ and $S(3)$ in Figures 1, 2 and the original $S(3)$ coloring didn't persist in the $S(4)$ coloring. This implies that we cannot directly work on the $S(5)$ coloring, but we should choose one random number (at least) and change its coloring.

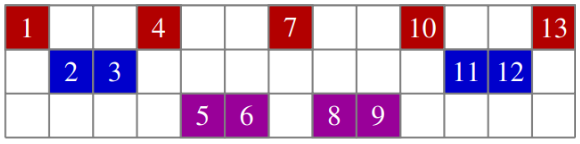


Figure 1: Schur Number 3 Coloring

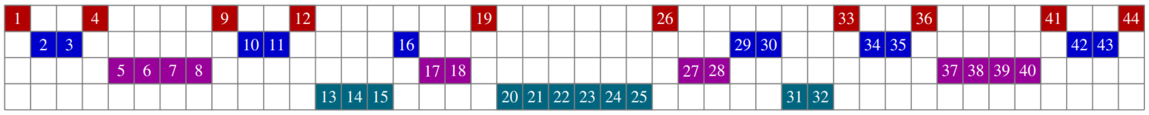


Figure 2: Schur Number 4 Coloring

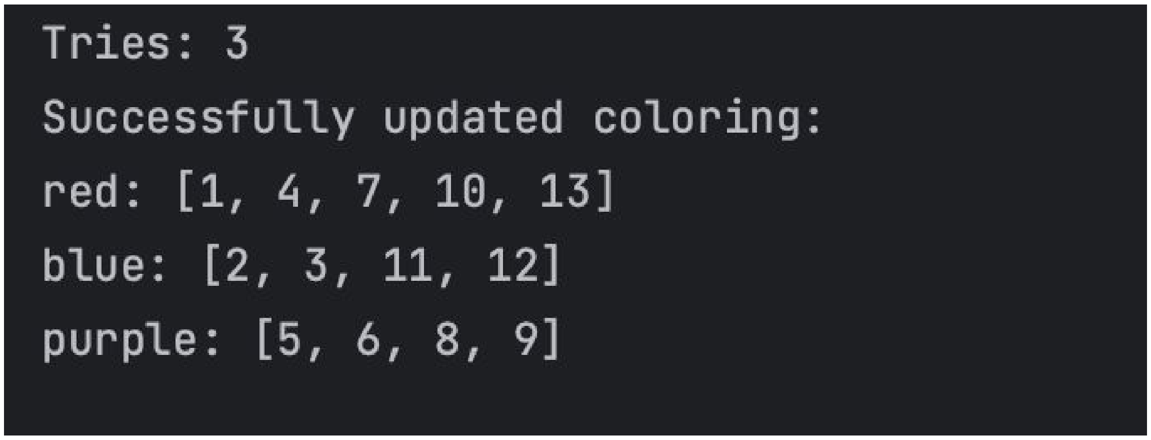
2.1 Randomness in Coloring

We would choose a random number and change its coloring. We designed an algorithm to test if changing number x from coloring A to coloring B is valid or not. If it is valid, we would then continue with the new coloring in the nested parallelism approach. If it is not valid, we would choose another random number and repeat the process. We would repeat this process until we find a valid coloring. The code for updating the coloring is included in the repository. The code takes as an input the amount of numbers that we wish to update its coloring randomly and return the updated coloring and how many steps it took to find a valid coloring.

2.2 Results

Here are the results of the recoloring of a coloring we made. We chose to recolor 1 number randomly. The results are as follows:

We also added a visualziation of how many tries it would take to do such random recoloring. The results are as follows:



```
Tries: 3
Successfully updated coloring:
red: [1, 4, 7, 10, 13]
blue: [2, 3, 11, 12]
purple: [5, 6, 8, 9]
```

Figure 3: Recoloring Results

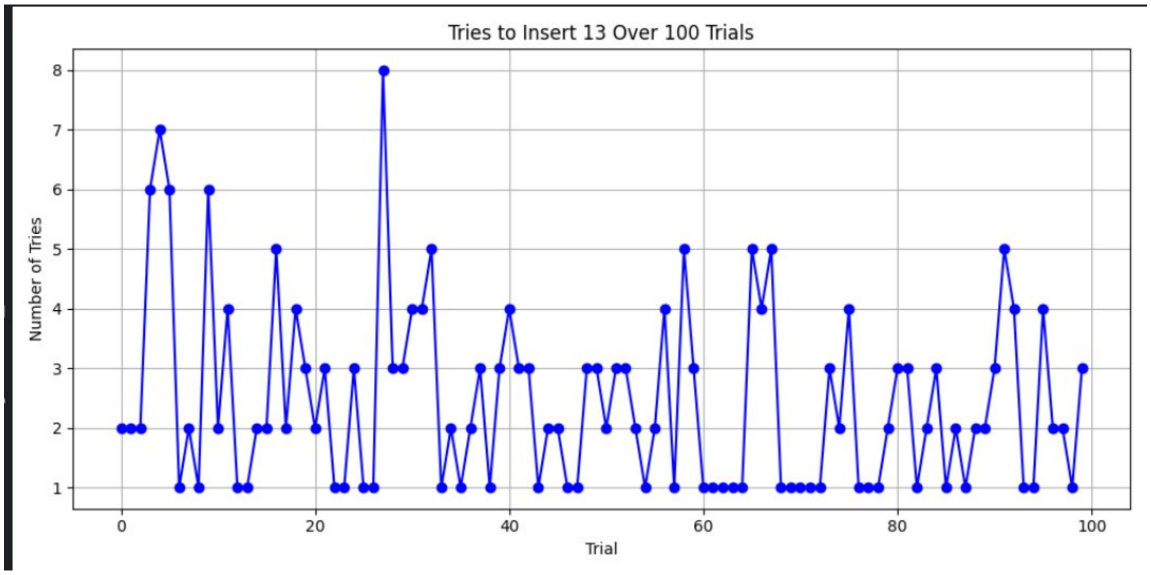


Figure 4: Number of random tries taken for 100 Trials of the Recoloring Algorithm