Assignment 3: Extra Credit

Justin Miller

April 5, 2012

Initial Approach

About

The first thing that I tried to do is check the creation of all the puzzles by finding relabling(row swap, band swap, column swap, pillar swap, and rotate) and relabling. From those two sets, let us call the first one α and the second one β, we know that $\forall \alpha_a, \alpha_b | \alpha_a ! = \alpha_b | a ! = b | a, b \in \mathbb{Z}$ and $\forall \beta_a, \beta_b | \beta_a ! = \beta_b | a ! = b | a, b \in \mathbb{Z}$ and $\exists a, b | a = b | a \in \alpha \, and \, b \in \beta$. So I know there will be at least one non-unique element if I calculate both $\alpha \, and \, \beta$.

So to check if something was unique, on creation we put into a list if that element is unique. To check if that element is unique we check the list to see if that element is in it. The list recreates itself at every 10,000 elements.

Memory

For this approach the memory is minimul. I never got a chance to see how much it actually took because of the time

Speed

The time to do this for α and β took 5 days, and that was parallel search. Maybe it would have been faster if as we put in the element in the list the list gets sorted, or it might have been slower.

Summary

For this approach, there has to be something faster.

Second Approach

About

This approach uses the same ideas as the first, but this time the data is put into a 2-3 tree. This way as an element is put in it is not only now in sorted order, but it is a balanced tree.

Memory

The memory takes more than the first approach. I also didn't get the issue fixed with the memory leak. So in the end this approach used 600mb in space.

Speed

This one is the fastest, the time to complete is 20 seconds while the vanilla without checking took 5 seconds.

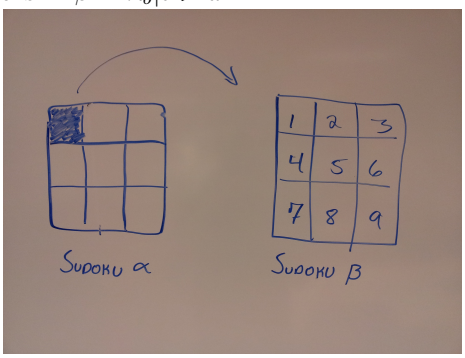
Summary

This found that there is one element that is not unique between the two which means that $\forall \beta_a | \lambda \notin perm(\beta_1) | \lambda \in perm(\beta_a) | a \geq 2$, or that the permutations of the rest of the of beta set are not in the permutation set of the first beta. But this doesn't calculate how many unique boards there are.

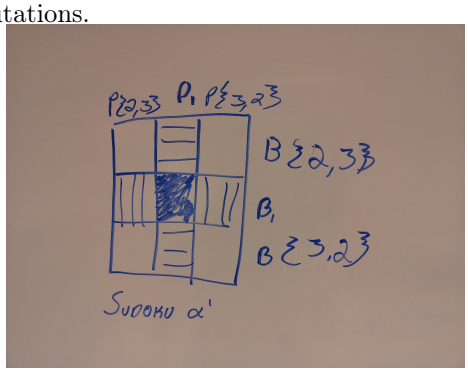
Future Approach

About

For this one, all we need to generate is the relable set, which we will call λ. From there we try and have, lets us say sudoku board $\alpha = \lambda_a$. And we are going to take the first 3x3 section in the upper left and go through each of 3x3 blocks in $\beta = \lambda_b | b > a$.



For each of those small sections, it will change the original 3x3 into the new one by using permutations. Then for the whole $\lambda^{'}$ we do those same changes. As well depending on where the new 3x3 goes we will do band and pillar permutations.



For the next part, those lines next to the filled in 3x3 represent locked in row/colum. We can then move those to fit the β. If it can't, then we know that the two are independent permutation sets. If we can, then we check the 3x3 that aren't filled in, but because of previous steps with all the columns, rows,

pillars, bands, and rotations locks in the last numbers. If they match, then we know that α and β are in the same permutation set.

Memory

This would use the least amount of memory of all the approaches.

Speed

This one will be on order $O(n^2)$ so it will be about $9!^2$ which based on the first approach we are going to estimate the time around a week.

Summary

This is going to be the only one that says how many unique boards there are.