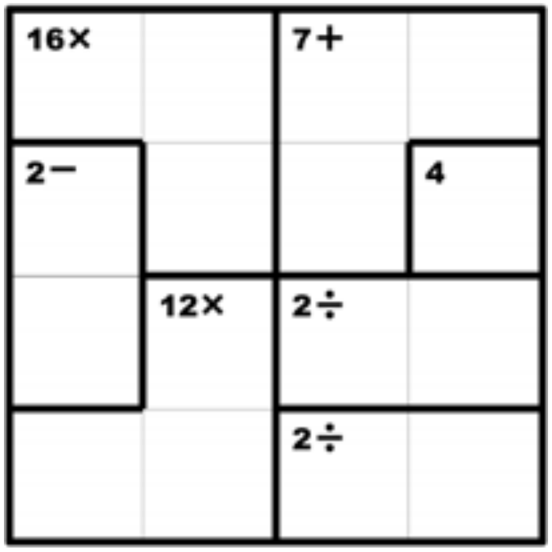
COMP 560

Assignment 1

Nick Egnatz



KenKen Puzzle Solver

Simple Backtracking Algorithm

The simple backtracking algorithm involved creating an empty 2d array of Nodes* of the puzzles size. In a while loop, it first checks if the current solution is valid using the validity checker* and if it is completely filled. If so, it returns the solved puzzle.

If not, it iterates through the values in each node in row major order. If it is no longer valid and every number has been tried for that node, it sets that node to 0 and tries another value in the previous node (backtracks). It follows this process until the solution is found, or there is no solution to the puzzle.

Best Backtracking Algorithm

The best backtracking algorithm is very similar in process to the simple backtracking algorithm, however, it uses domain filtering to limit the amount of iterations needed to complete the puzzle. It does this by checking if the value that is about to be tested for validity is already guaranteed to fail the validity checker. If so, that value is removed from the domain for that node. This speeds up the simple backtracking algorithm and lessens the amount of iterations and checks to solve the puzzle.

Local Search Algorithm

The local search algorithm uses the hill climbing technique to solve the given KenKen puzzle. It uses a point system to improve the current solution and starts with a random solution with no repeats in the rows. It selects a random row and column and swaps that value with another value in that same row. If then tests if the point* value for the new solution is higher than the previous. If it is, that becomes the new solution and the algorithm continues until a solution is found or it reaches a maximum.

Contributions

I was responsible for the entirety of the assignment. The solutions, helper functions, and other material needed.

**Node*
A node is a class that has two variables, a current state (integer value) and a maximum state (the length of the puzzle)

**Validity Checker*
For all algorithms, a validity checker is used to determine whether or not the tried value works or not. This is a 2 step function that checks if there are duplicates in the rows and columns (ignoring 0s) and it also uses the checkMath function to determine if the math is correct for each block (ignoring blocks that contain 0s)

**Points*
The point system is calculated by testing how many cages have correct values and how many rows/columns satisfy the constraints. Points increase when the solution is getting closer to a valid solution.