Algorithm: Backtracking of Sudoku board

```
function ValidateBoard(board,x,y)
    if DuplicateNumbersinRow(board,x)
        return false
    if DuplicateNumbersinCol(board,y)
        return false
    if DuplicateNumbersinBox(board,x,y)
        return false
    return true


function Solve(board,unAssignInd,N_unAssign)
    if N_unAssign==0    // No more empty position, solution found
        return true
    index=unAssignInd[N_unAssign]
    for val in [1, BoardSize]
      {
        board[index.x,index.y]=val   // Set guess
        if ValidateBoard(board,index.x,index.y)
            {
              // Solve recursively
              sol=Solve(board,unAssignInd,N_unAssign-1)
              if sol return true
            }
      }
    board[index.x,index.y]=0  // No solution, reset value in backtracking
    return false
```

Parallelism: Checking one guess is an independent parallel task but two tasks can not work on the same board at the same time. When the solution is found, no more tasks should be issued and previous tasks cancelled. Note, tasks can be created inside tasks to create enough work for the threads in the task queue but too small tasks should be avoided (due to memory and task overhead).

Boards: You are provided with three different boards to solve. The data in the files are stored as 1 byte integers. The first number is the base (e.g. 5) , the second number is side length (e.g. 25) , and following numbers are the board numbers.