Improved Recursive BIsection

* Compute N0 and N1 to T0 and T1
  + First T0 = floor(½)
  + Next, compute ratio = (float) T0 / T
  + Then No = (int)(n \* ratio)

Solving Maze Puzzle

* To make this problem more concrete, let’s consider a maze represented by a matrix of characters. An exaple 6x6 input maze is (a 2D char array)

Logic for Solution

FIND-PATH(x,y)

* if(x, y) outside maze
  + Return false
* If (x,y ) is goal
  + Return true
* if( x,y not open )
  + Return false
* If (x,y has been visited)
  + Return false
* Mark (x,y) as part of solution path
* if(FIND-PATH(North of x,y) == true)
  + Return true
* if(FIND-PATH(East of x,y) == true)
  + Return true
* if(FIND-PATH(South of x,y) == true)
  + Return true
* if(FIND-PATH(West of x,y) == true)
  + Return true
* Unmark x,y as part of solution path
* Return false

S#####

. . . . . #

#. ####

#. ####

. . . #. G

##. . . #

Sudoku Puzzle

* Sudoku ( A Constraint satisfaction problem) can also be solved using backtracking.
* Let us Look at a 4 by 4 simplified Sudoku as an example, called Shidoku

1 . | . .

. 2 | . .

\_\_ |\_\_

. . | 3 .

. . | . 4

Based on the input, we can easily calculate the candidate numbers for each cell. For example, Cell(row 1, colum 2) can be number 3 or 4.)