

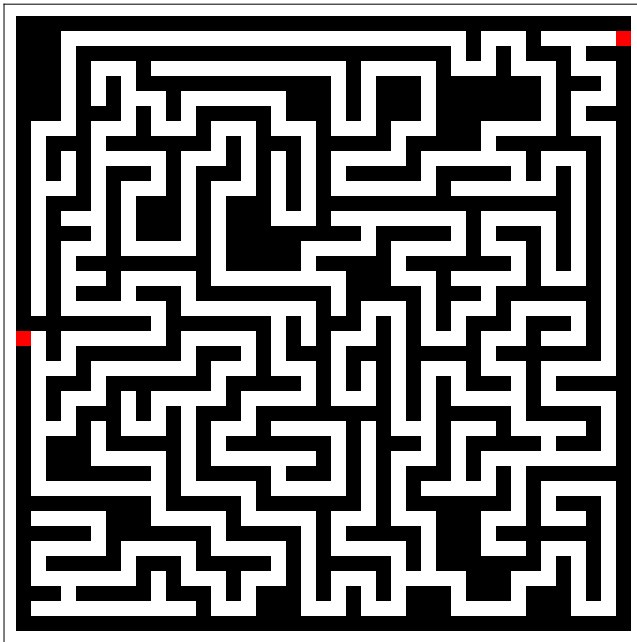
Maze Program

(c) Matsuda@symbolics.jp 2010.5.7

displaying a maze

```
In[13]:= showMaze[board_] :=  
  ArrayPlot[Reverse[board], ColorRules -> {e -> Red, 1 -> Black, 0 -> White}]
```

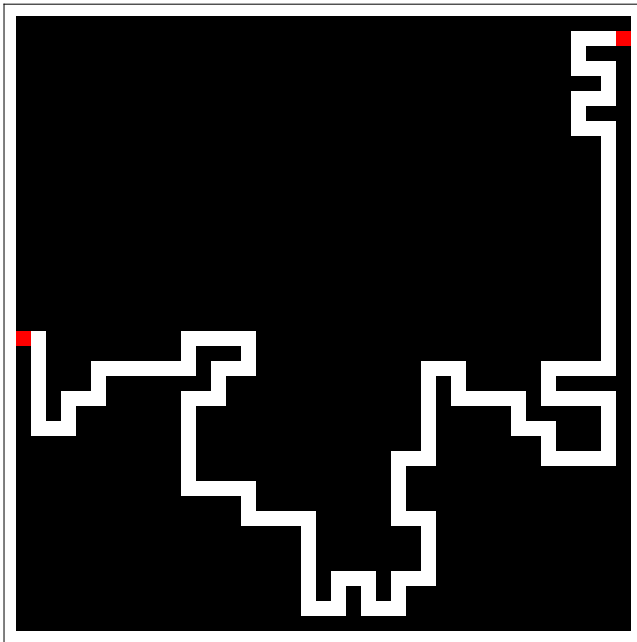

In[15]:= showMaze[maze]



serial version

```
In[16]:= findPathOfMaze[board_] := Module[{mazeSolve, vnNeighbors0},
  (* the following update rules can be replaced with CellularAutomaton[] *)
  mazeSolve[0, 1, 1, 1, 0] := 1;
  mazeSolve[0, 1, 1, 0, 1] := 1;
  mazeSolve[0, 1, 0, 1, 1] := 1;
  mazeSolve[0, 0, 1, 1, 1] := 1;
  mazeSolve[0, 1, 1, 1, 1] := 1;
  mazeSolve[x_, _, _, _, _] := x;
  Attributes[mazeSolve] = Listable;
  vnNeighbors0[maze_] :=
    Map[RotateRight[maze, #] &, {{0, 0}, {0, 1}, {1, 0}, {0, -1}, {-1, 0}}];
  updateMaze[maze_] := mazeSolve @@ vnNeighbors0[maze];
  FixedPoint[updateMaze, board]
]
```

```
In[17]:= findPathOfMaze[maze] // showMaze
```



```
Out[17]=
```

parallel version

The following program is relatively NOT faster than the serial version because problem size is relatively small and the communication overhead is large.

This program demonstrates that parallel-CA approach can be easily applied to with MPI programming.

```
In[18]:= padding[mat_] := Module[{m}, m = Table[0, {Length[mat] + 2}, {Length[mat[[1]]}];
  m[[2 ;; -2]] = mat;
  m];

In[19]:= peal[mat_] := Module[{m, n},
  {m, n} = Dimensions[mat];
  mat[[2 ;; m - 1]]];

In[20]:= parallelFindPathOfMaze[board_, n_] := Module[{mazeSolve, vnNeighbors0, mat},
  mazeSolve[0, 1, 1, 1, 0] := 1;
  mazeSolve[0, 1, 1, 0, 1] := 1;
  mazeSolve[0, 1, 0, 1, 1] := 1;
  mazeSolve[0, 0, 1, 1, 1] := 1;
  mazeSolve[0, 1, 1, 1, 1] := 1;
  mazeSolve[x_, _, _, _, _] := x;
  Attributes[mazeSolve] = Listable;
  vnNeighbors0[maze_] :=
  Map[RotateRight[maze, #] &, {{0, 0}, {0, 1}, {1, 0}, {0, -1}, {-1, 0}}];
  updateMaze[maze_] := mazeSolve @@ vnNeighbors0[maze];

  (* note: FixedPoint is not used because of making the algorithm be simpler *)
  mpiScatter[board, mat, 0, mpiCommWorld];
  mat = padding[mat];
  Nest[Function[{a}, EdgeCell[mat]; mat = updateMaze[mat]], mat, n];
  Flatten[peal[mat], 1]
]
```

```
In[21]:= parallelFindPathOfMaze[maze, 218] // showMaze
```

```
Out[21]=
```

