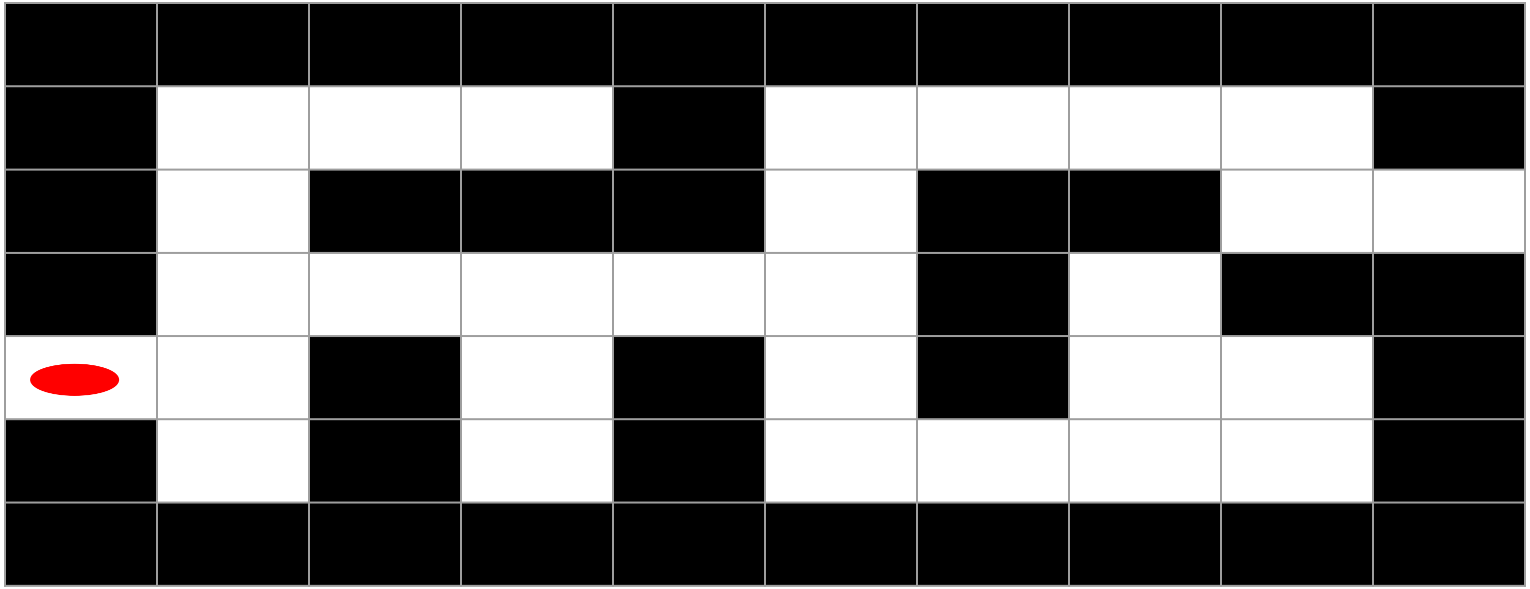
MAZE

CSC 236

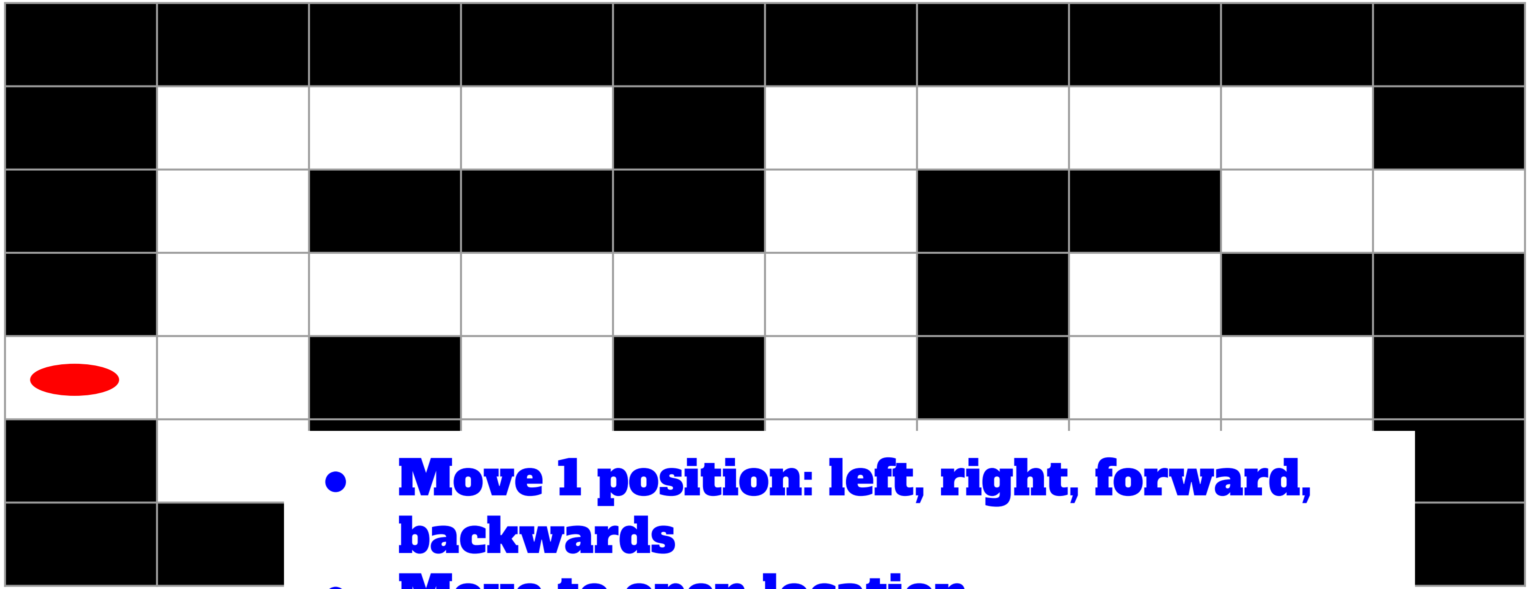
MAZE

- Read the specification on the web site
- See calendar for due date
- Optional team assignment
  - Can work with one other student (2-person team)
  - Both must submit assignment

Navigate a maze

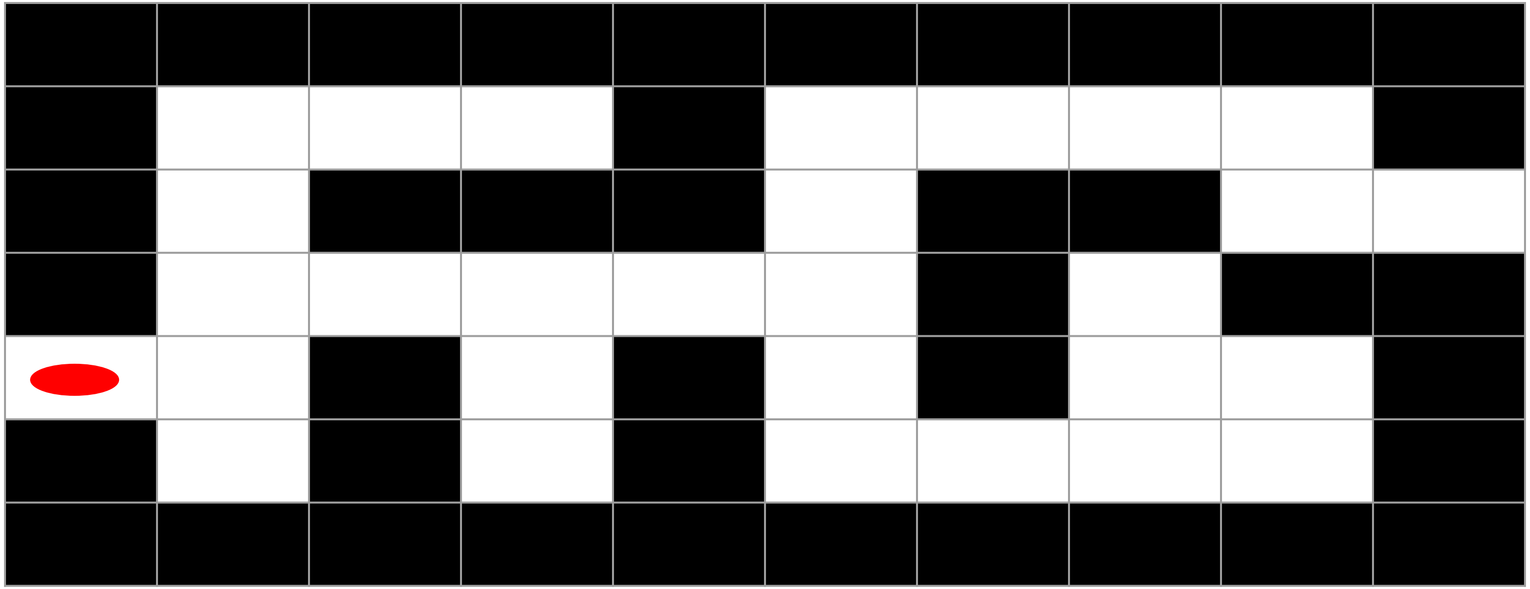


Navigate a maze

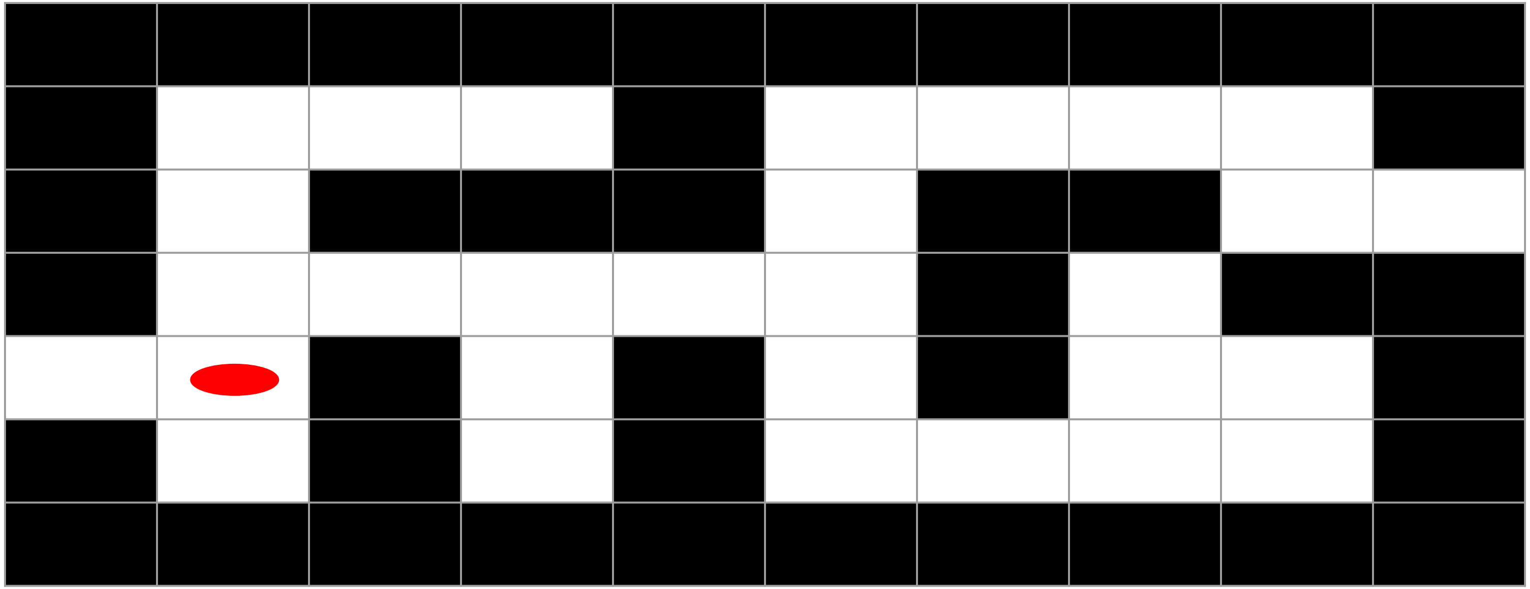


- Move 1 position: left, right, forward, backwards
- Move to open location
- North & South boundaries are blocked
- East & West blocked except for entry & exit

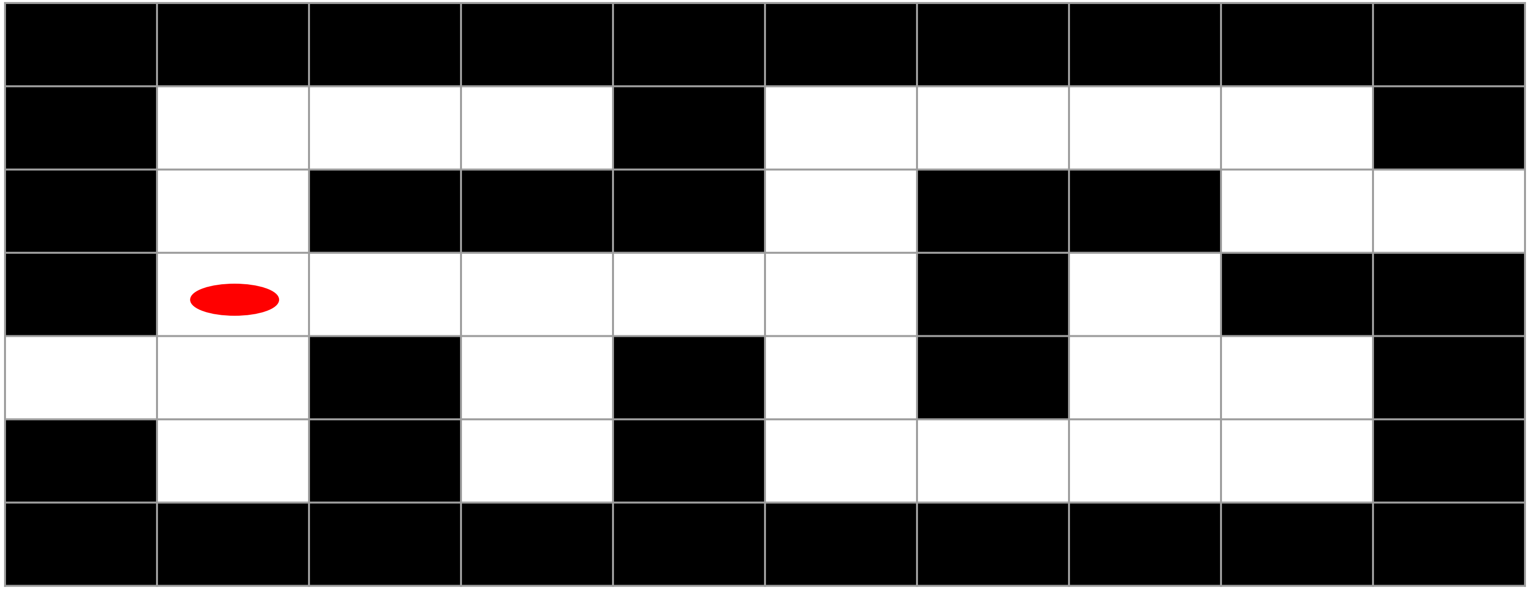
Navigate a maze



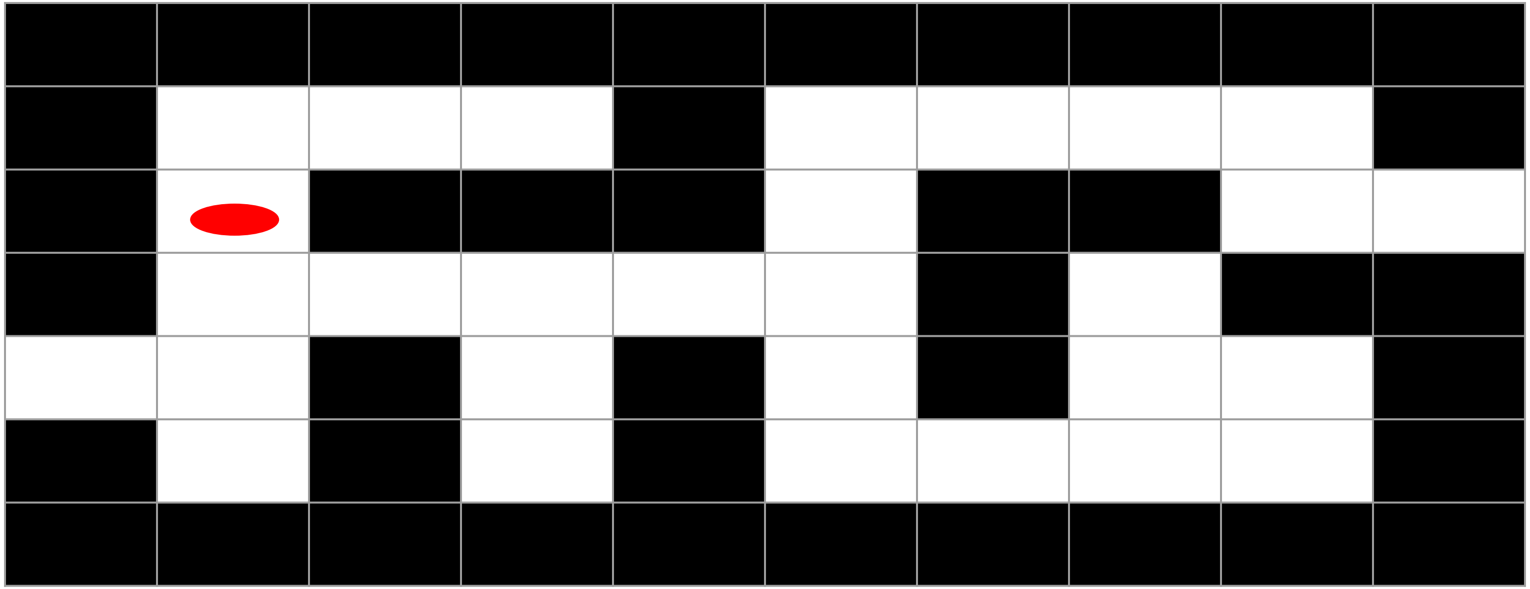
Navigate a maze



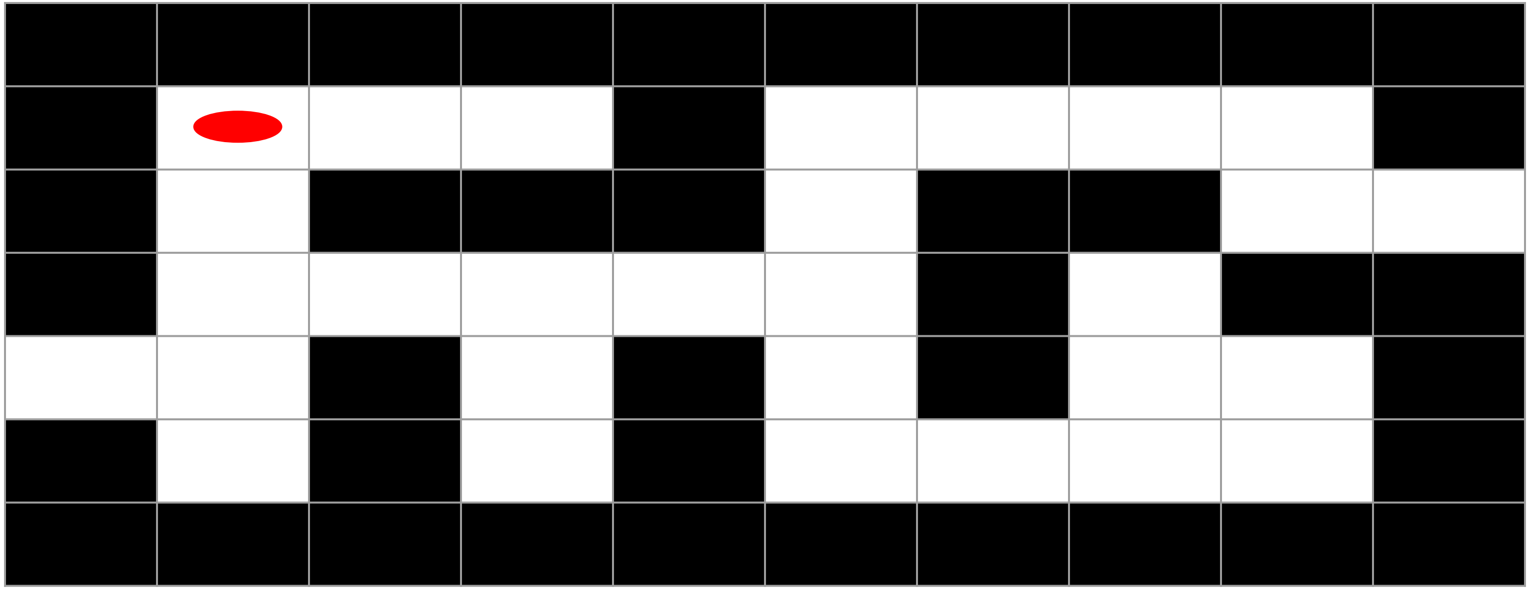
Navigate a maze



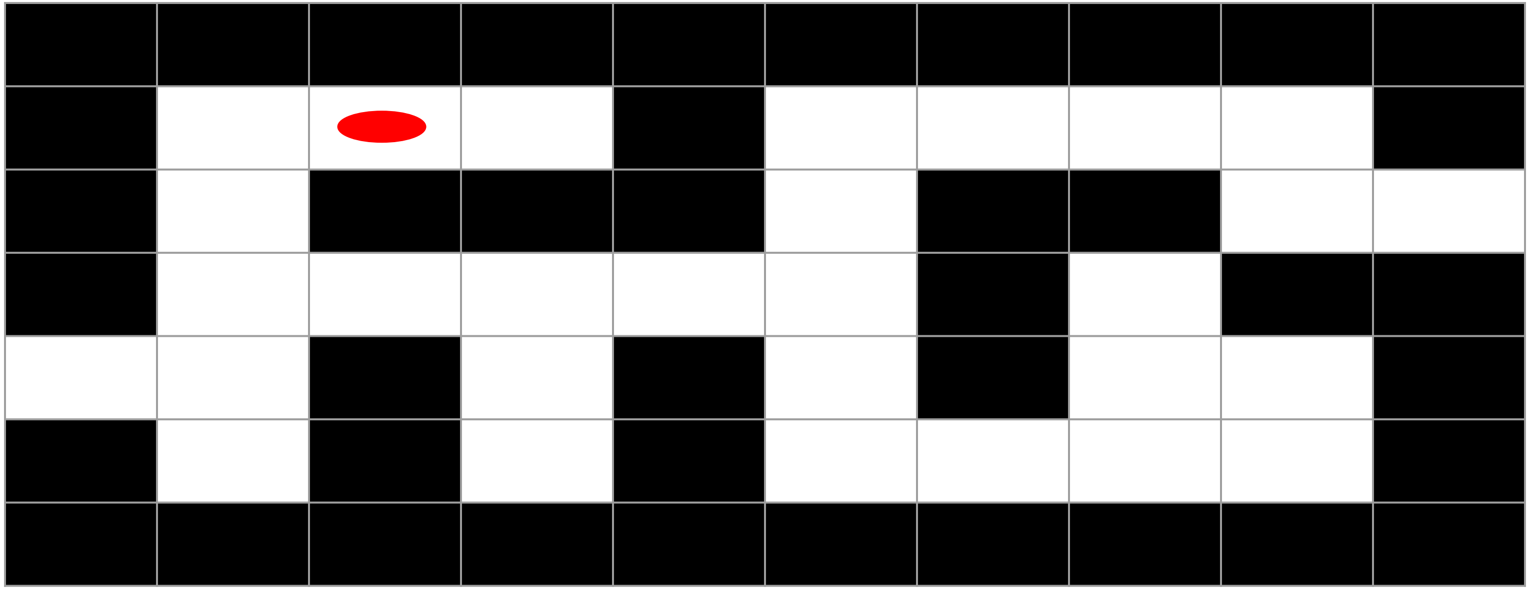
Navigate a maze



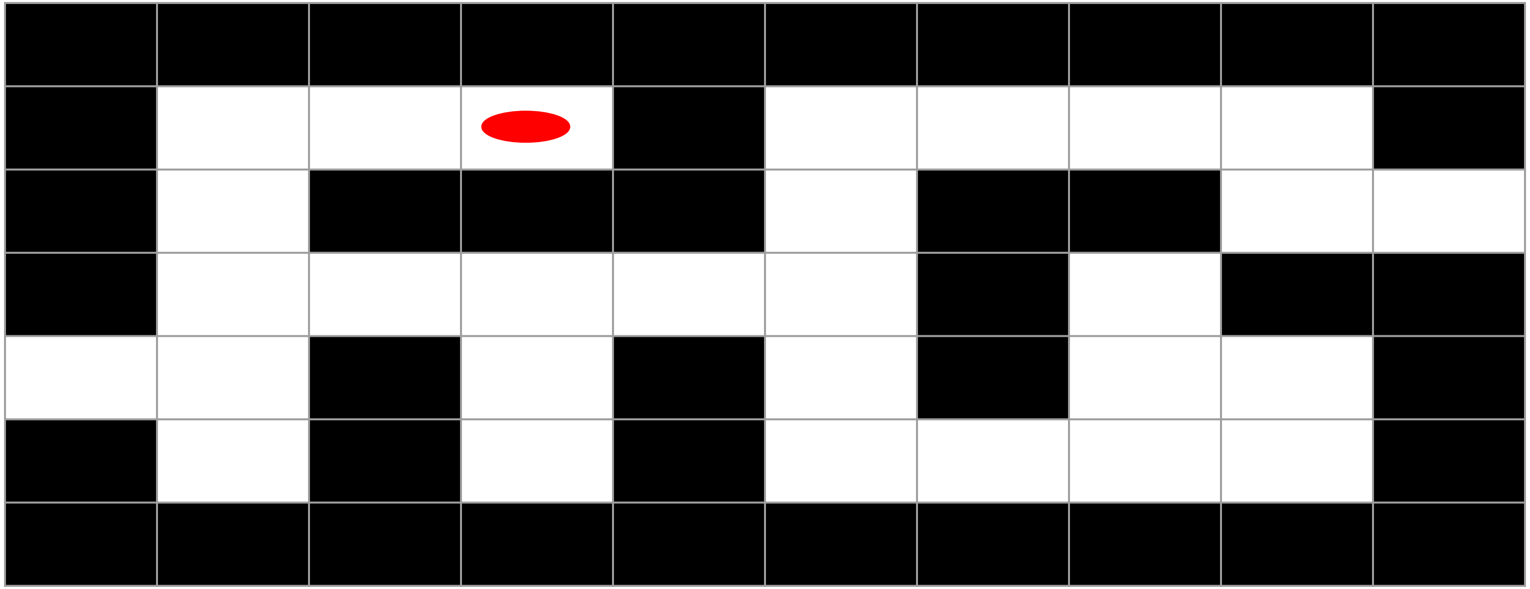
Navigate a maze



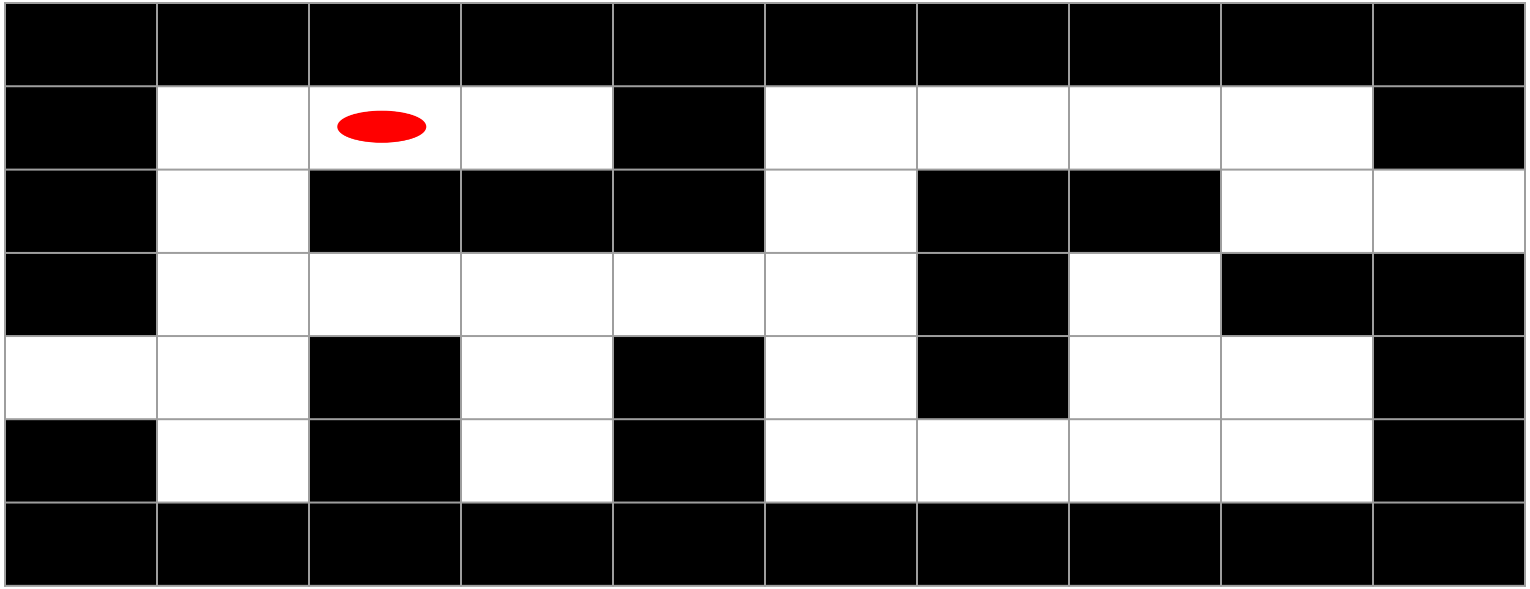
Navigate a maze



Navigate a maze

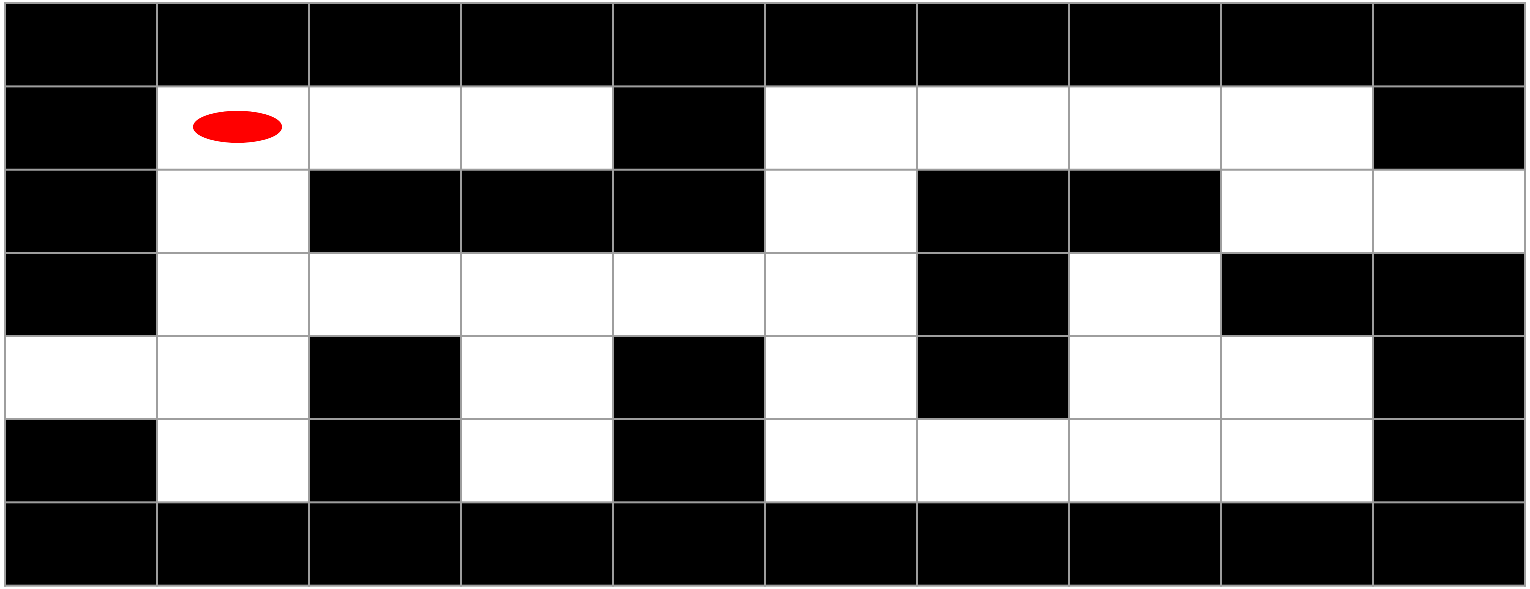


Navigate a maze

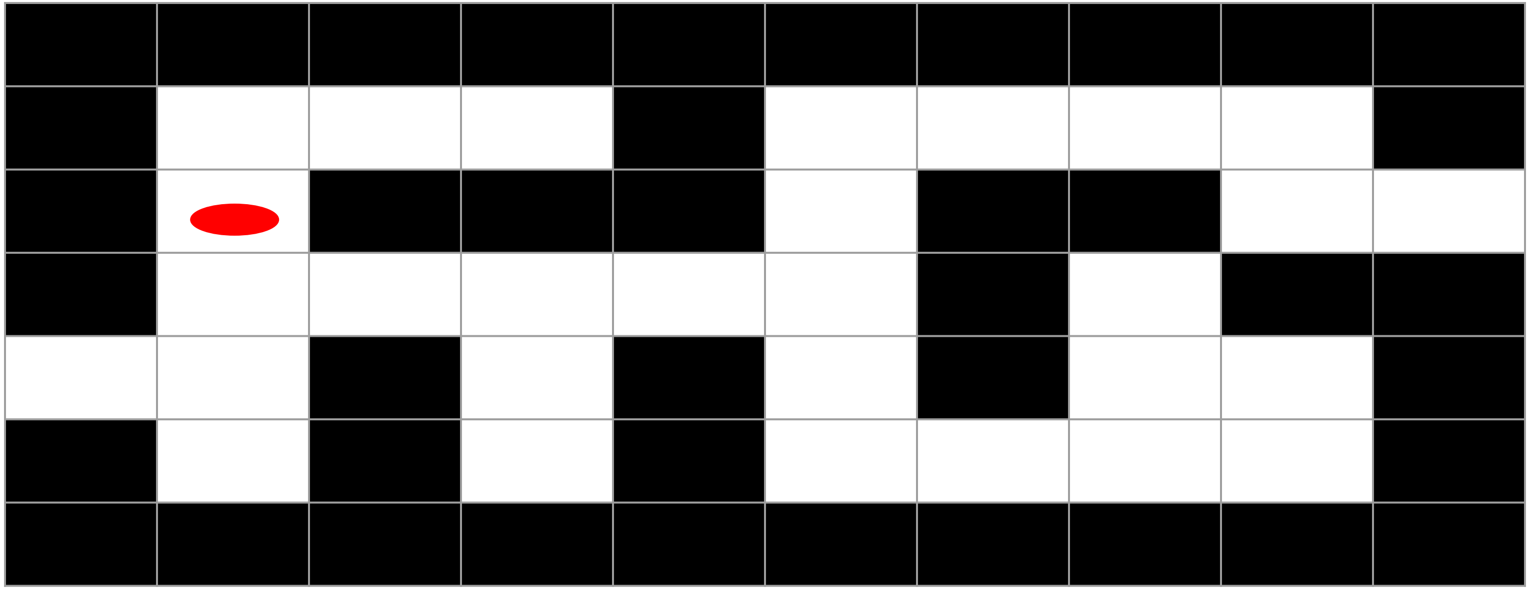


■	■	■	■	■	■	■	■	■	■
■	□	●	□	■	□	□	□	□	■
■	□	■	■	■	□	■	■	□	□
■	□	□	□	□	□	■	□	■	■
□	□	■	□	■	□	■	□	□	■
■	□	■	□	■	□	□	□	□	■
■	■	■	■	■	■	■	■	■	■

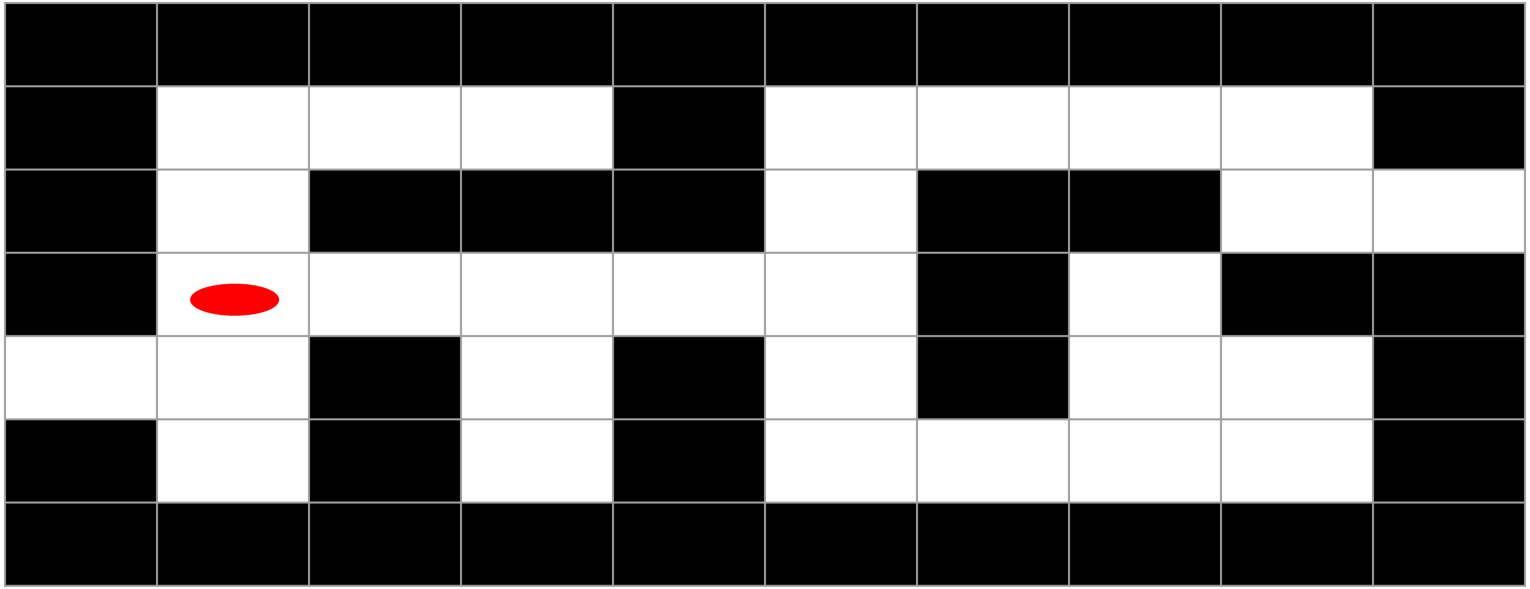
Navigate a maze



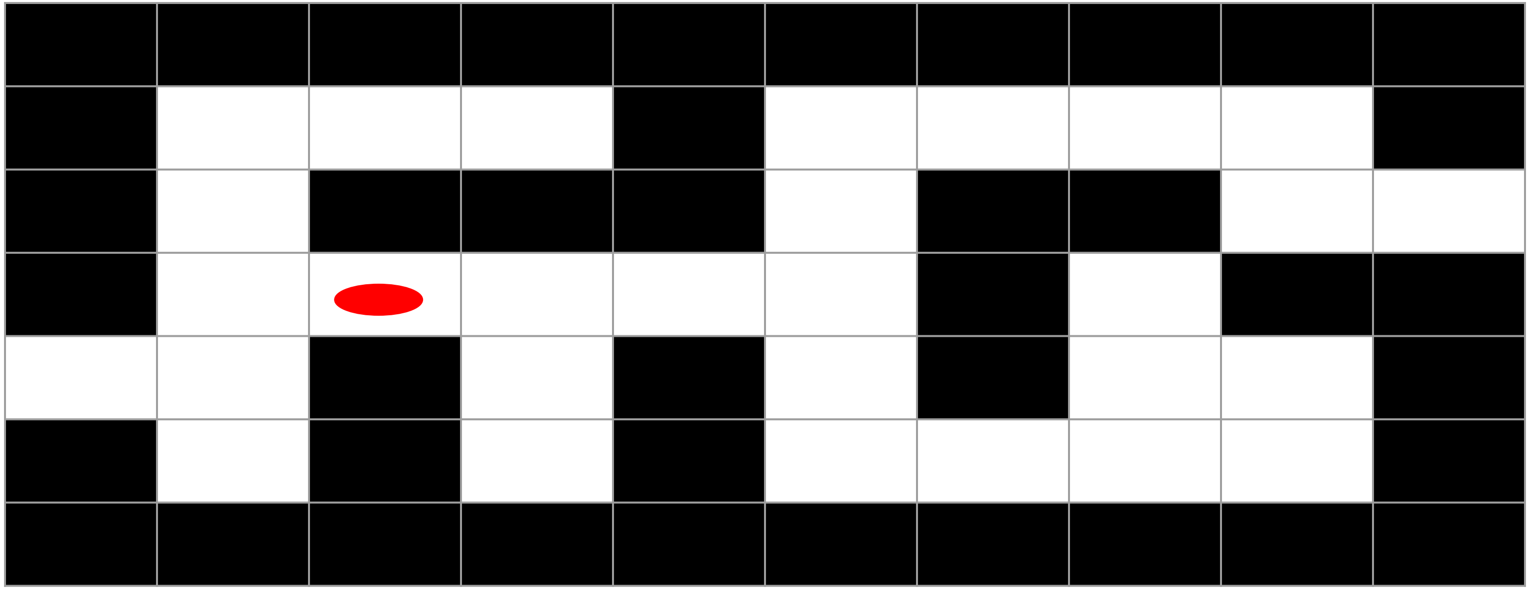
Navigate a maze



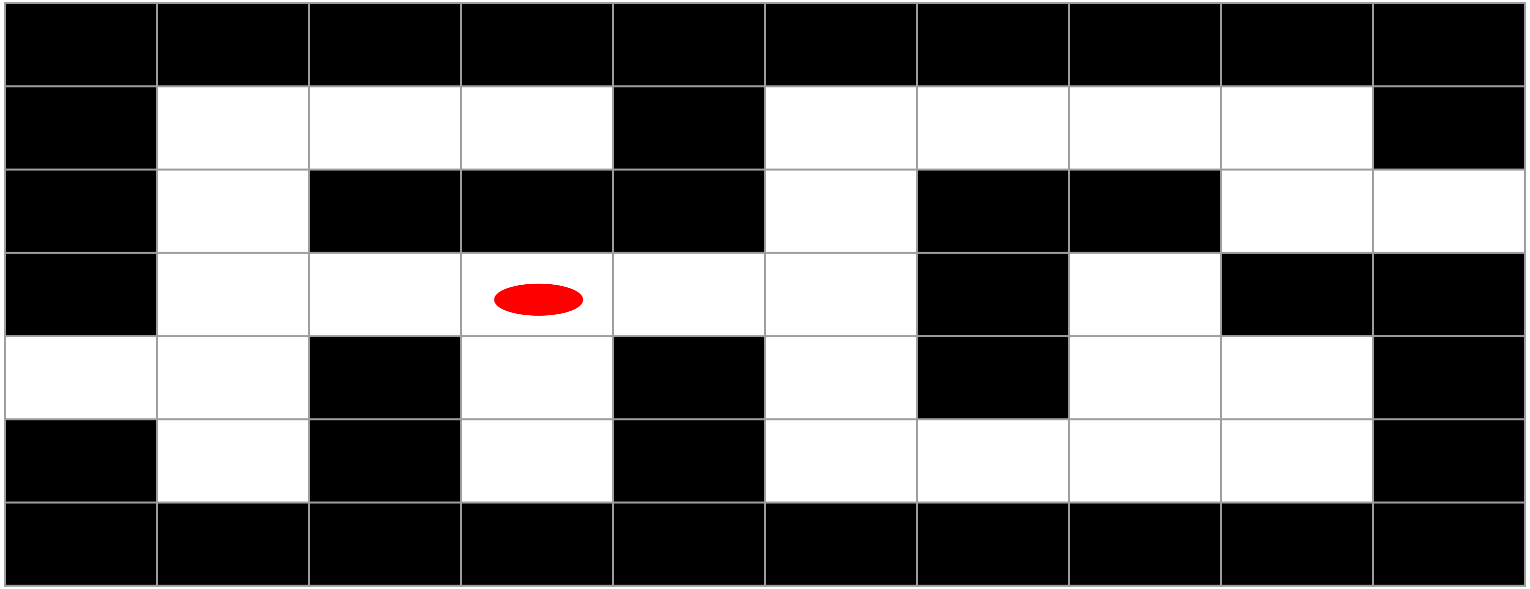
Navigate a maze



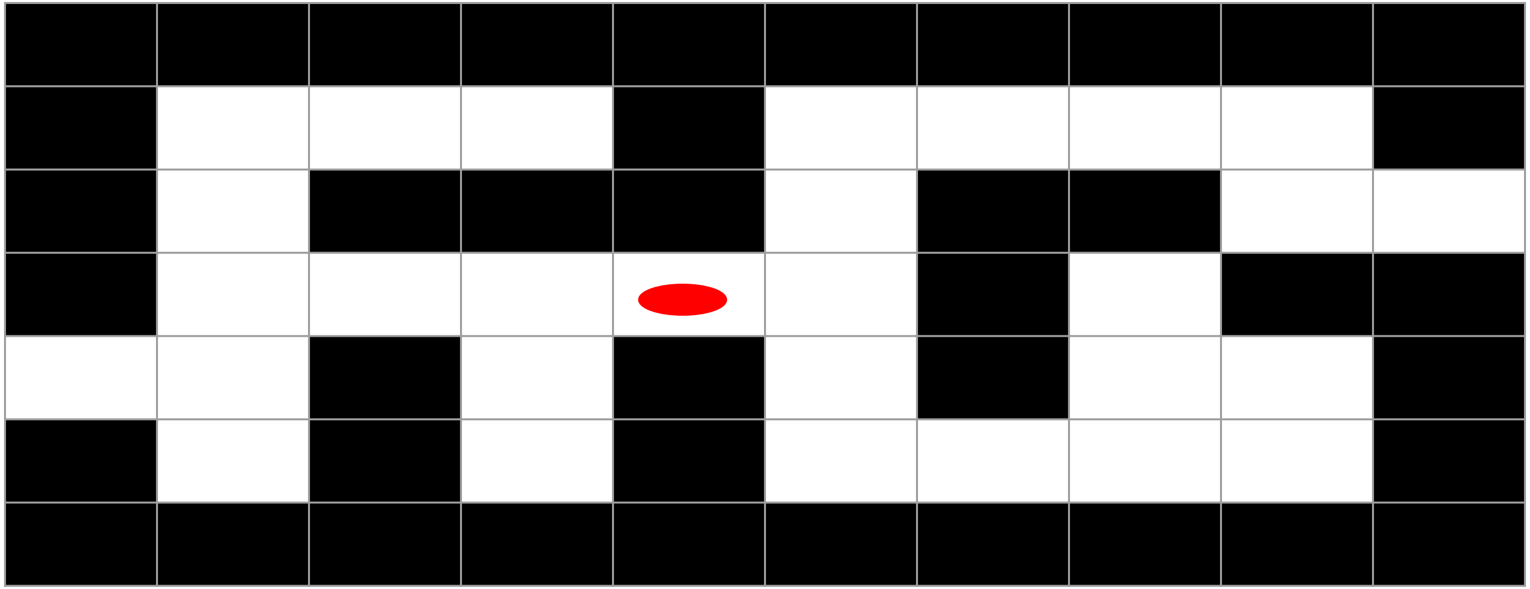
Navigate a maze



Navigate a maze

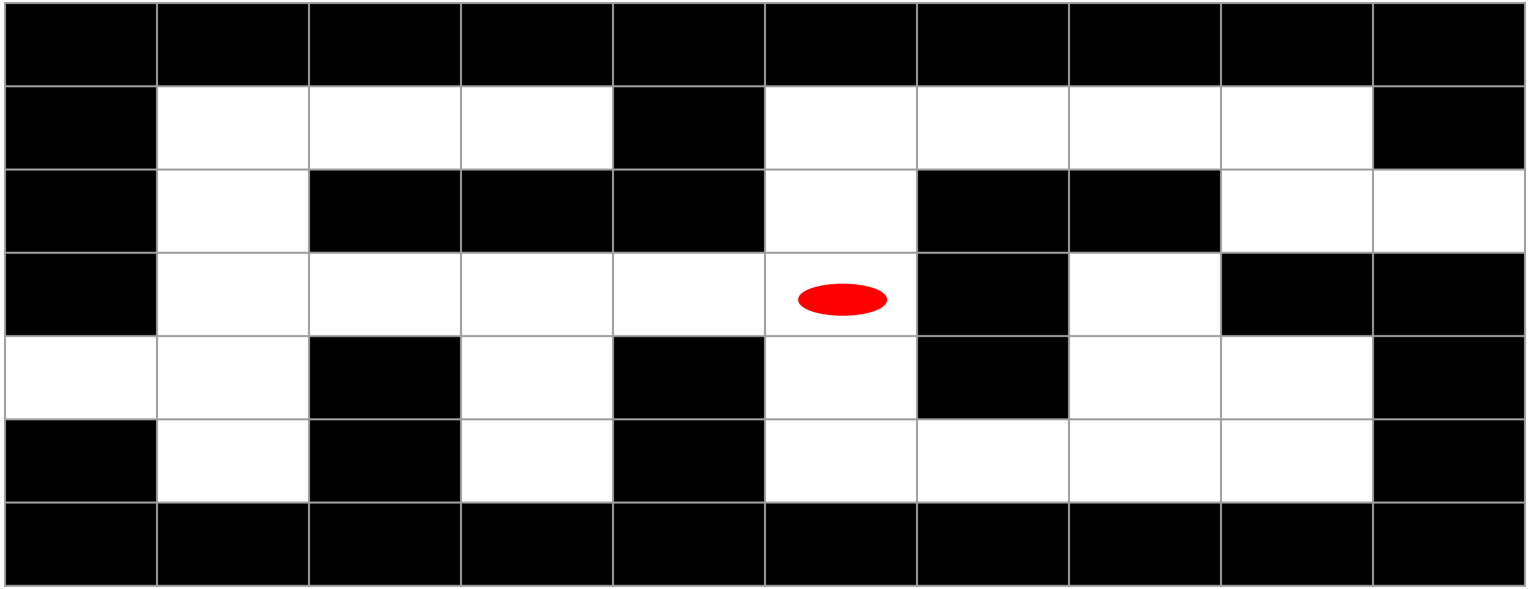


Navigate a maze

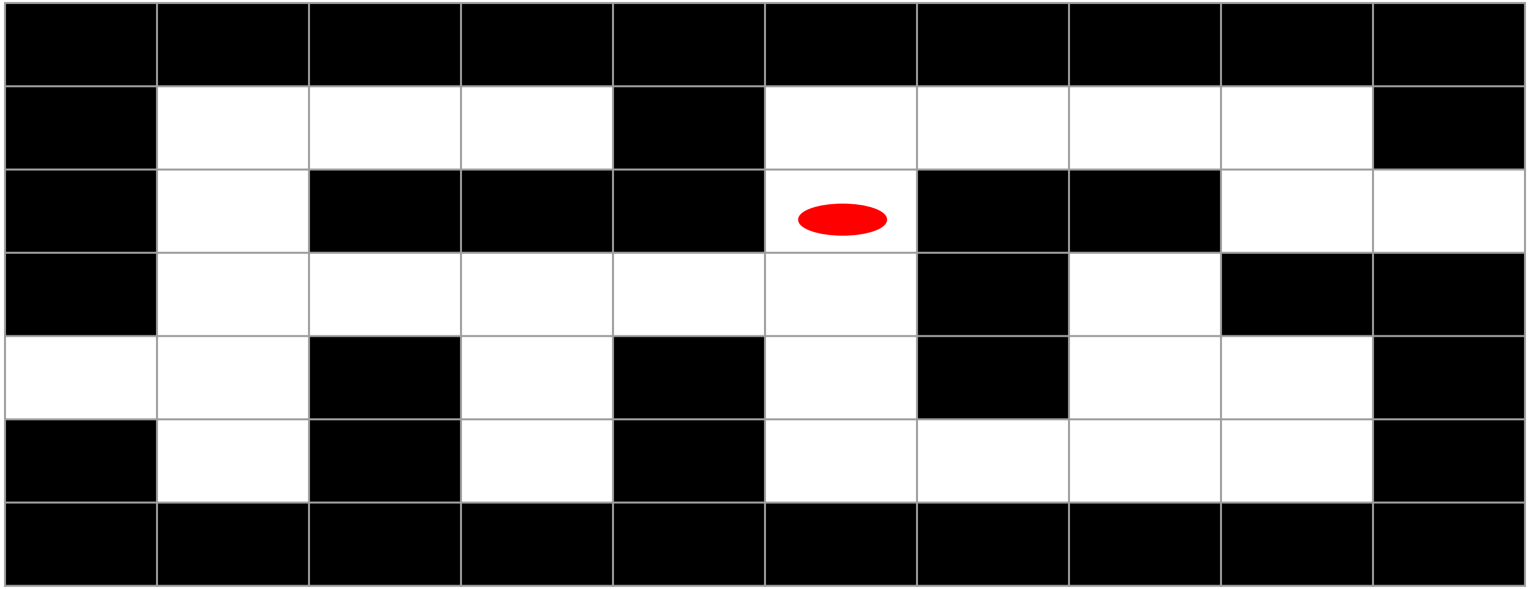


■	■	■	■	■	■	■	■	■	■
■	□	□	□	■	□	□	□	□	■
■	□	■	■	■	□	■	■	□	□
■	□	□	□	●	□	■	□	■	■
□	□	■	□	■	□	■	□	□	■
■	□	■	□	■	□	□	□	□	■
■	■	■	■	■	■	■	■	■	■

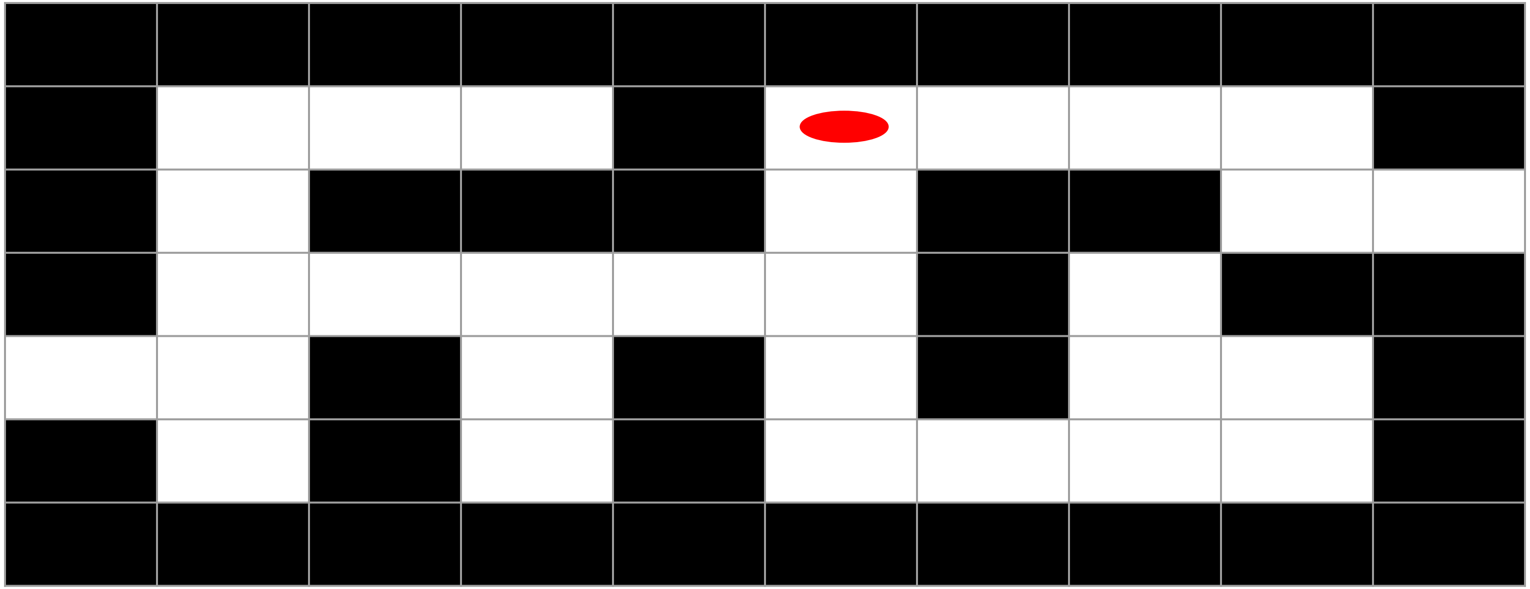
Navigate a maze



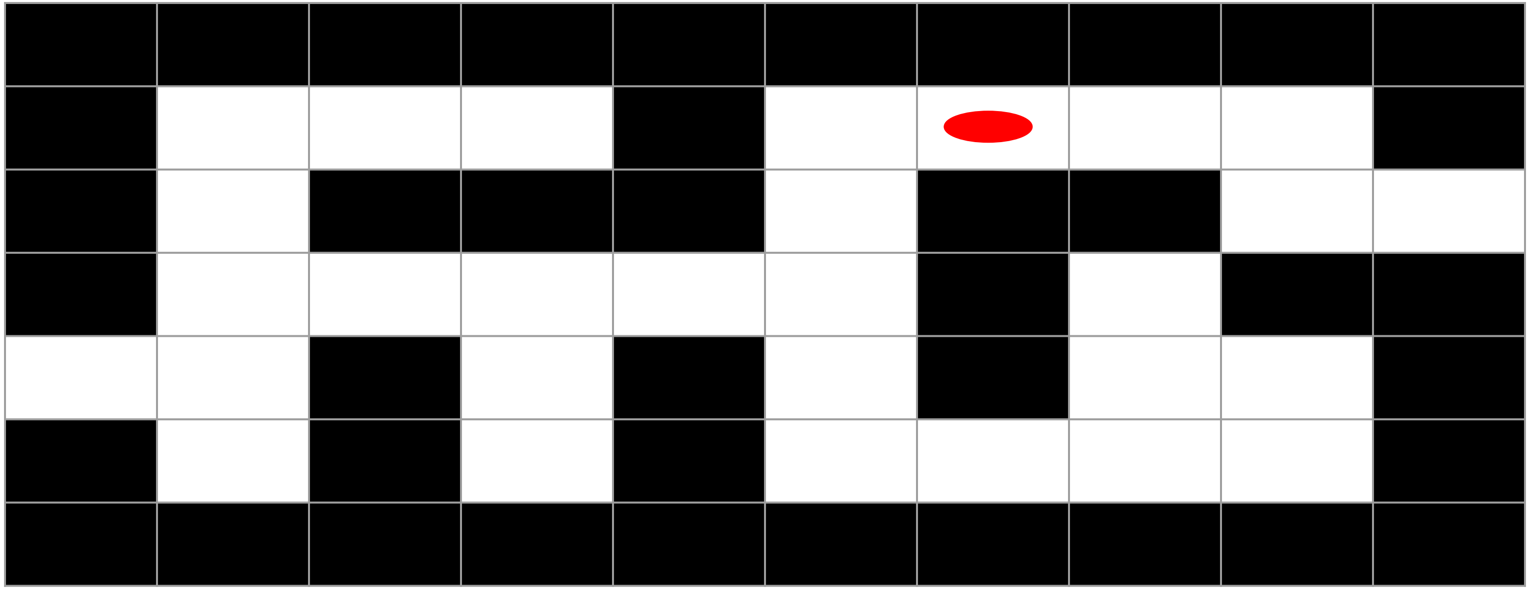
Navigate a maze



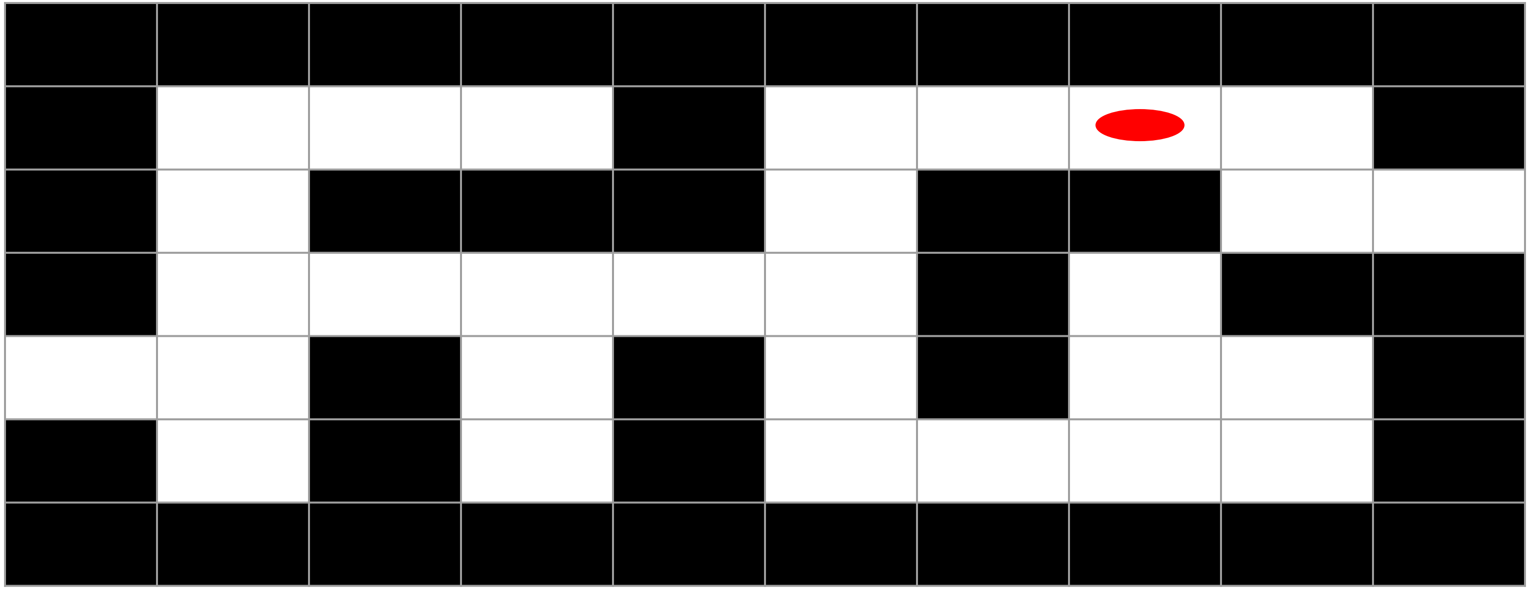
Navigate a maze



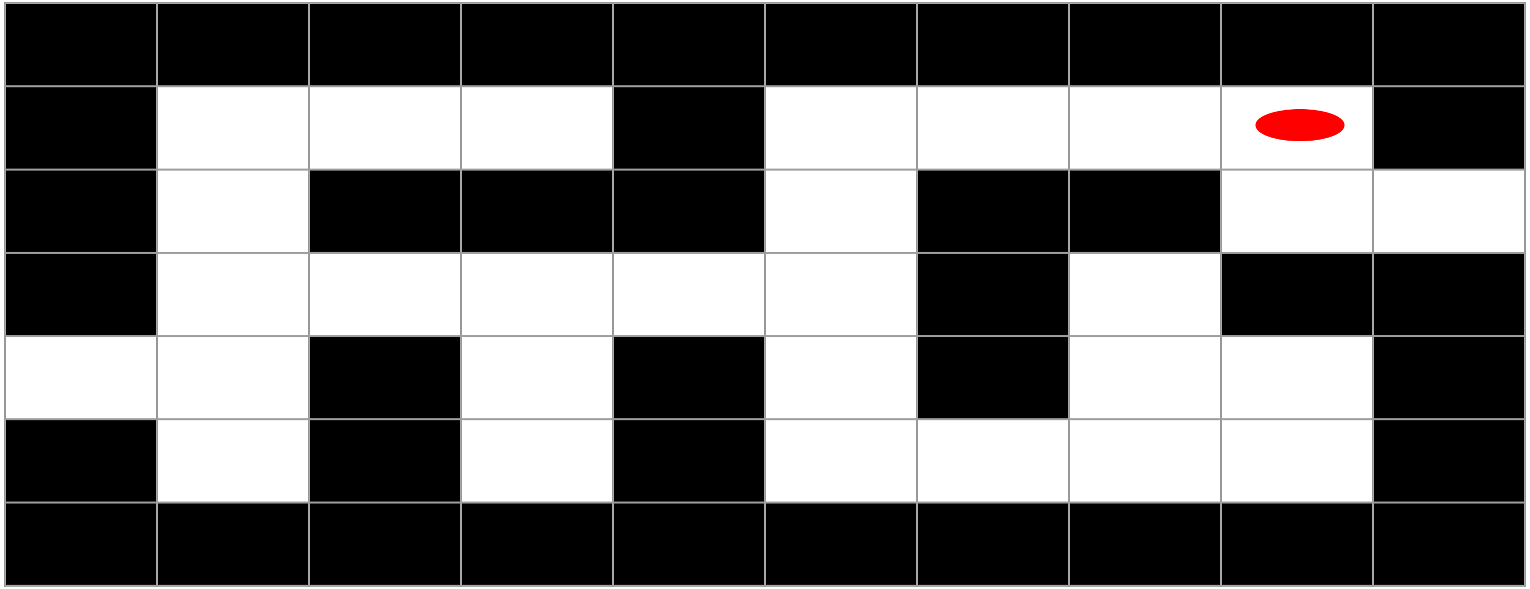
Navigate a maze



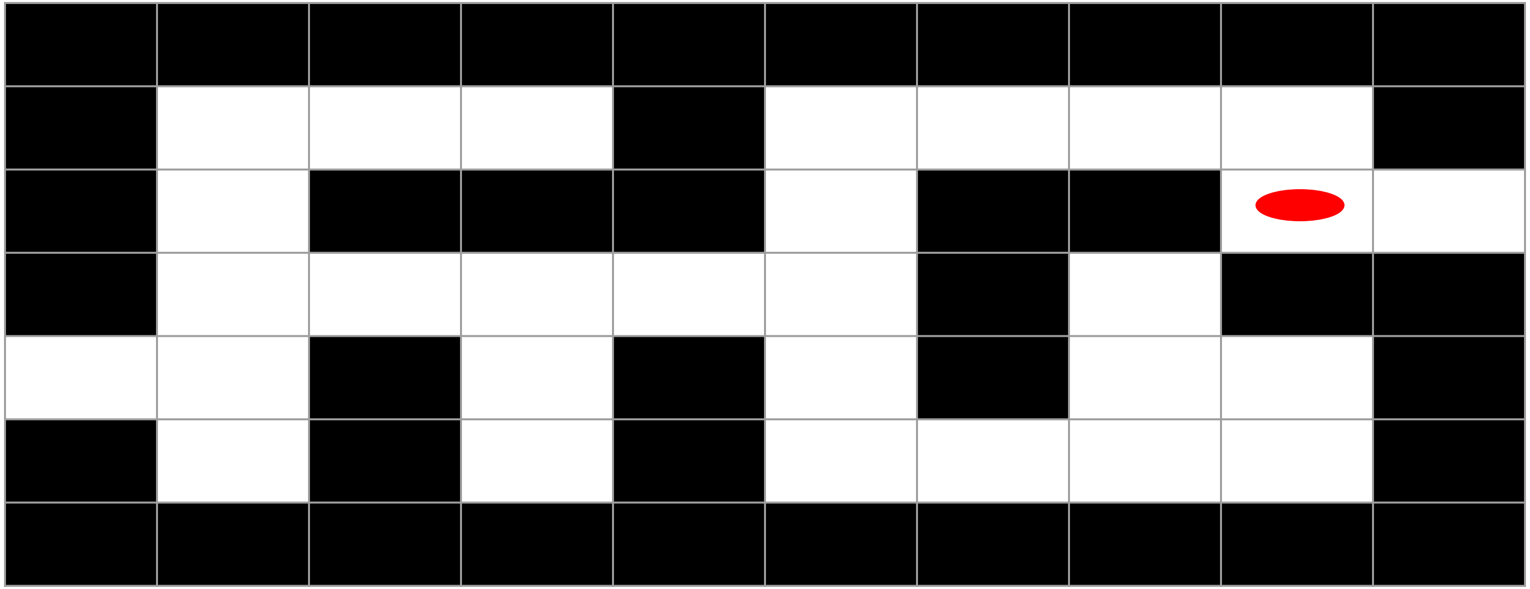
Navigate a maze



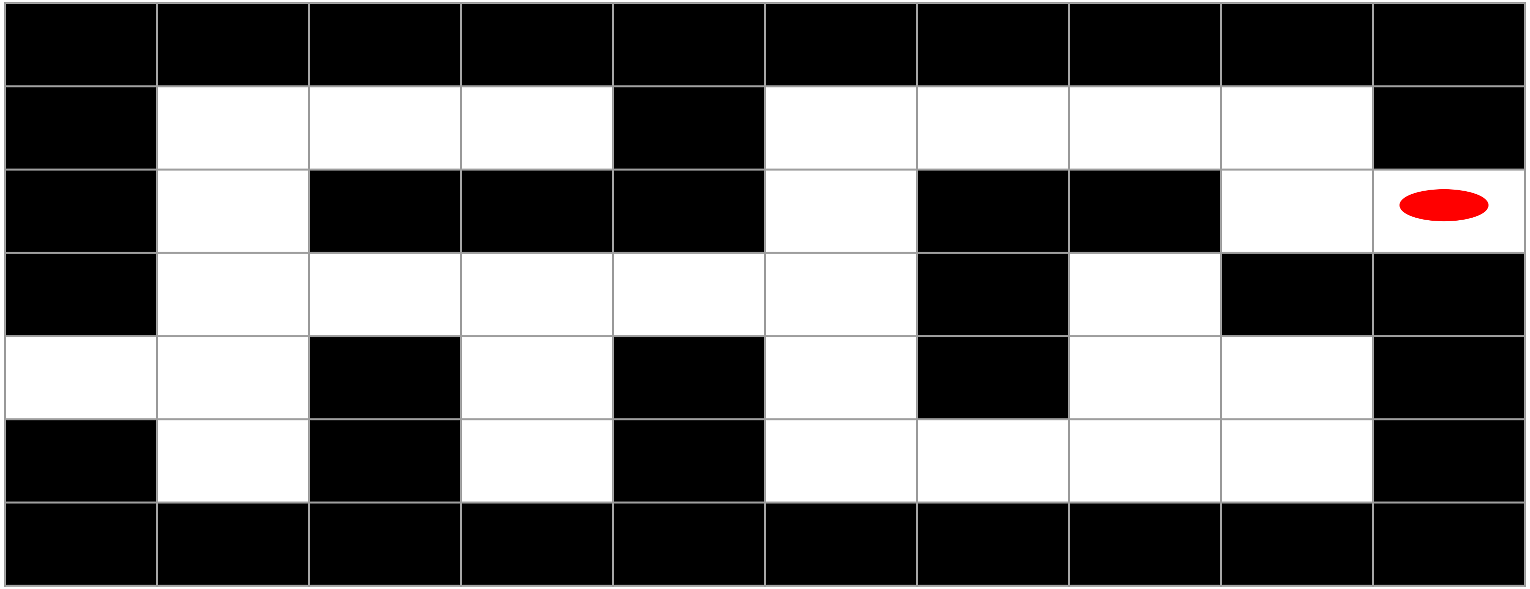
Navigate a maze



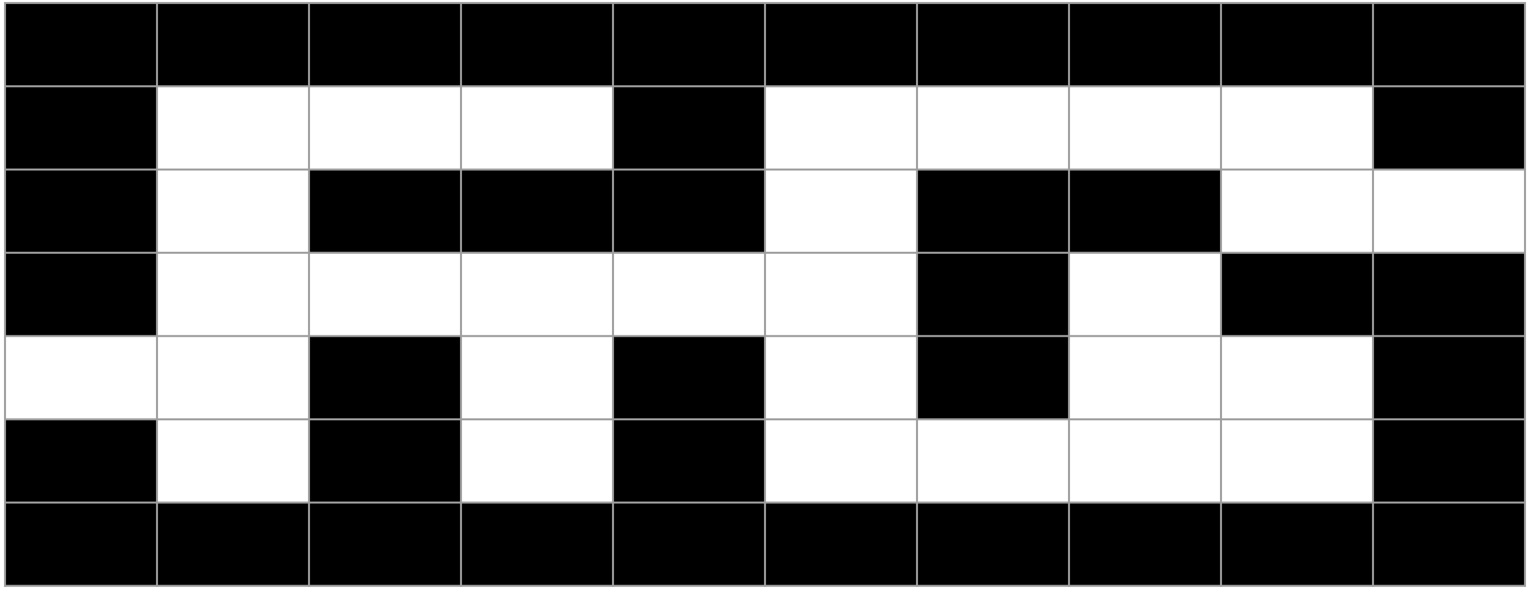
Navigate a maze



Navigate a maze



Navigate a maze



Subroutine

- Guides mouse through 15x30 maze
- Create subroutine
    - File: nextval.asm
    - Entry: nextval
- Return next move
- Four parameters -- subroutine has no memory

Parameters

- Global (non-local) data
- Accessed via pointers (address)
  - Parameters are addresses -- not the data itself
  - Use indirect addressing

Parameters

- bp — address of the maze
- di — address of current y value
  - Current location of mouse N/S (up/down)
  - Range 1 to 15
- si — address of current x value
  - Current location of mouse E/W (right/left)
  - Range 1 to 30
- bx — address of current direction of travel
  - Unsigned byte (E=1, S=2, W=3, N=4)

No error checking of input is required

Action

- Determine mouse’s next move
- Update data
  - Update Y (di)
  - Update X (si)
  - Update direction (bx)

X or Y (never change both)

Notes

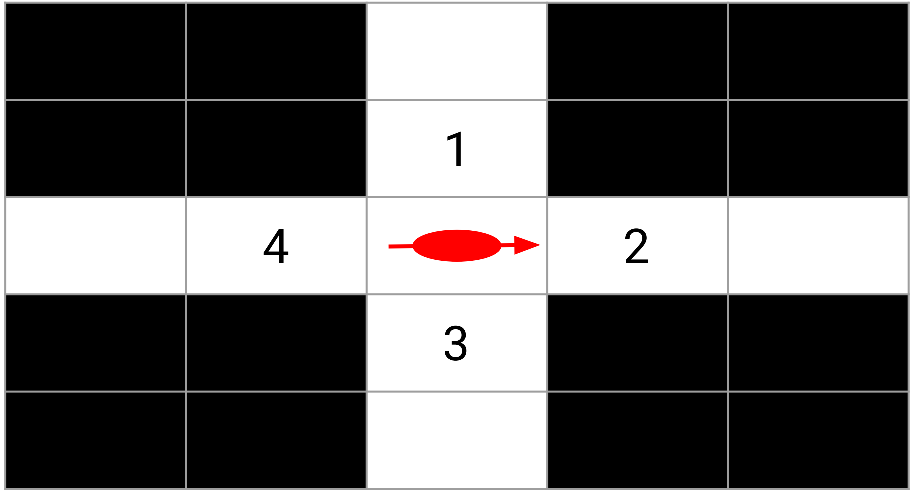
- no error checking is needed
- driver detects if mouse traversed maze
- mouse is not allowed to stay in a square
- mouse may not return to start square
- do not do any file I/O
- do not keep history info between calls
- do not modify the maze
- mouse must work for any 15 x 30 maze

Algorithm

- “Turn left”
  - Mouse takes the left-most turn possible
  - It will eventually exit the maze

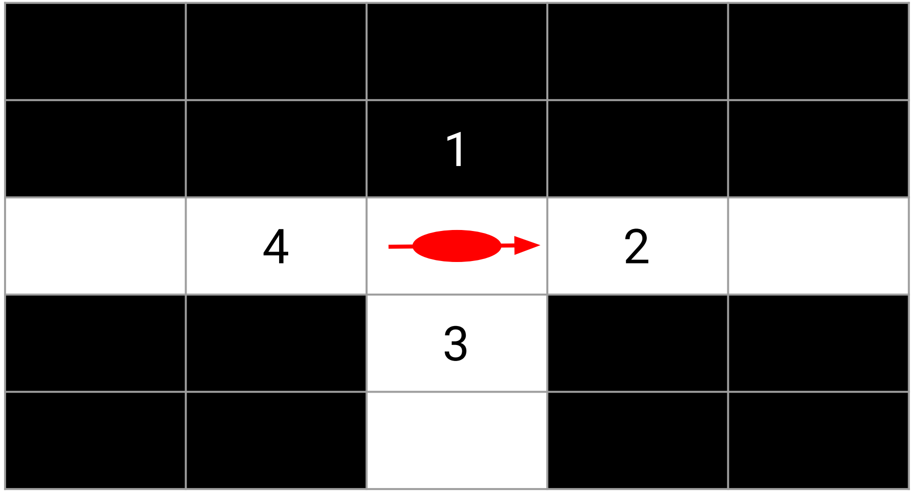
Algorithm

- “Turn left”
  - Mouse takes the left-most turn possible
  - It will eventually exit the maze
- Try to turn left



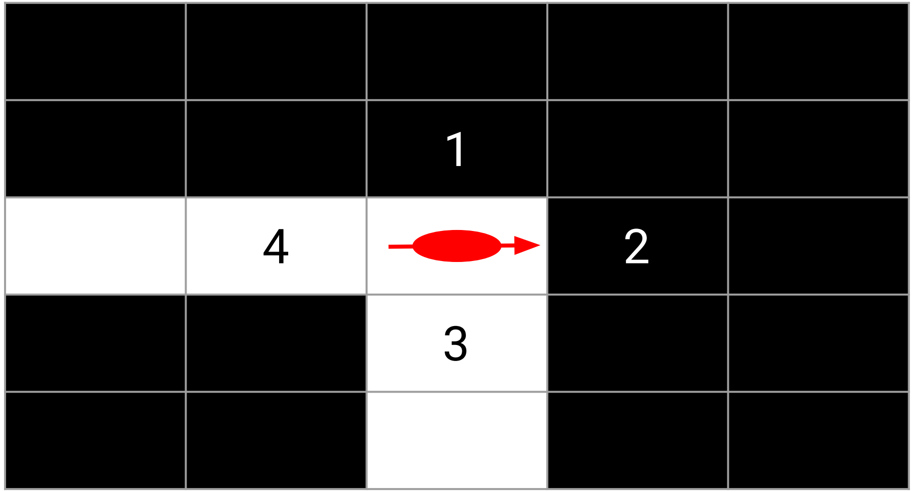
Algorithm

- “Turn left”
  - Mouse takes the left-most turn possible
  - It will eventually exit the maze
- Try to turn left
- Else try forward



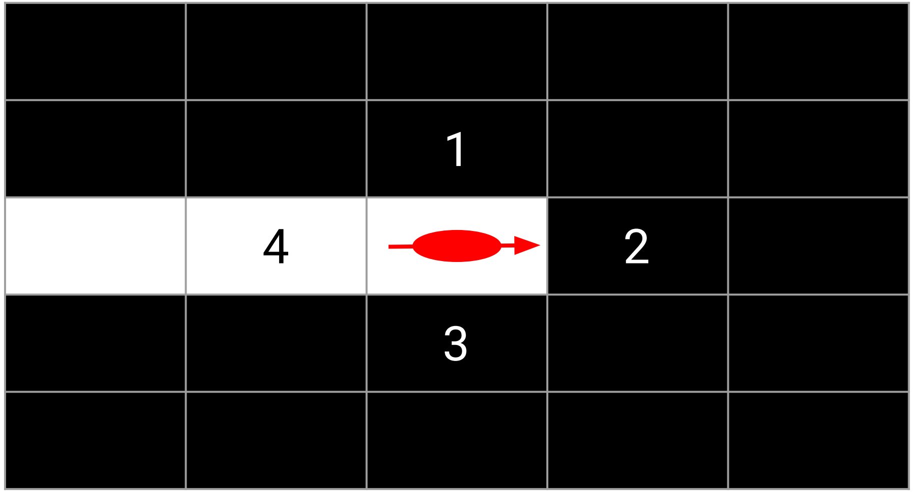
Algorithm

- “Turn left”
  - Mouse takes the left-most turn possible
  - It will eventually exit the maze
- Try to turn left
- Else try forward
- Else try right



Algorithm

- “Turn left”
  - Mouse takes the left-most turn possible
  - It will eventually exit the maze
- Try to turn left
- Else try forward
- Else try right
- Else go backwards
  - This must work



Algorithm

- “Turn left”

- Try to turn left
- Else try forward
- Else try right
- Else go backwards
  - This must work

- “Turn right”

- Try to turn right
- Else try forward
- Else try left
- Else go backwards
  - This must work

How to access 2-D array data

- Memory is 1-D
- A 2-D is flattened in memory

Logical view

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

Physical view

0	1	2	3	4	5	6	7	8	9	10	11
1,1	1,2	1,3	1,4	2,1	2,2	2,3	2,4	3,1	3,2	3,3	3,4

How to access 2-D array data

- Memory is 1-D
- A 2-D is flattened in memory
- Offset
  - (y-1) x width + (x-1)

Logical view

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

Physical view

0	1	2	3	4	5	6	7	8	9	10	11
1,1	1,2	1,3	1,4	2,1	2,2	2,3	2,4	3,1	3,2	3,3	3,4

How to access 2-D array data

- Memory is 1-D
- A 2-D is flattened in memory
- Offset
  - (row-1) x width + (column-1)
  - (2-1) x 4 + (3-1)

Logical view

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

Physical view

0	1	2	3	4	5	6	7	8	9	10	11
1,1	1,2	1,3	1,4	2,1	2,2	2,3	2,4	3,1	3,2	3,3	3,4

offset

How to access 2-D array data

- Memory is 1-D
- A 2-D is flattened in memory
- Offset
  - (row-1) x width + (column-1)
  - (3-1) x 4 + (2-1)

Logical view

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

Physical view

0	1	2	3	4	5	6	7	8	9	10	11
1,1	1,2	1,3	1,4	2,1	2,2	2,3	2,4	3,1	3,2	3,3	3,4

offset

Relative addresses

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

- Know the mouse location
  - Say (2,2)
  - Know offset as well
- Access above
  - One fewer row
  - `current = ( row   -1) x width + (column-1)`
  - `above   = ((row-1)-1) x width + (column-1)`
  - `diff    =  - width`

Relative addresses

- Above: `- width`
- Below: `+ width`
- Left: `- 1`
- Right: `+ 1`

Steps

- Retrieve unpack.exe from maze locker
- nextval
  - nextval.m is the model for your subr
  - rename nextval.m to nextval.asm
  - all source code must be in nextval.asm
- mazedrvr.obj is the driver program
  - link your nextval.obj with mazedrvr.obj
  - creates the executable — mazedrvr.exe

Driver program

- Driver program
  - Reads a “maze” file
  - Builds and displays maze
  - Displays mouse
  - Calls `nextval`
  - Moves mouse
  - Checks for completion or error (eg, moved onto blocked square)
- Test program
  - `testmaze maze.nn`
  - 6 maze programs are provided

Grading

- 50% — correct
- 20% — documentation
- 15% — instructions written
- 15% — instructions executed

- Submit: “maze.ans”