Math 124 - Programming for Mathematical Applications

UC Berkeley, Spring 2024

Project 2 - Random Maze

Due Friday, March 1

Description

In this project, you will write a computer code to generate a random maze using a recursive algorithm. You will also write a code to find a path between two points in a given maze.

The integer `n` specifies the size of the `n`-by-`n` array of cells in the maze. Note the matrix indices `i,j` specify the x and y-coordinates, respectively (see plot below).

The horizontal and the vertical *interior* walls of the maze are described by the arrays:

- `H`, Bool array of size `n`-by-`n–1`
- `V`, Bool array of size `n–1`-by-`n`

These arrays specify if there is a wall or not between two neighboring cells.

An example is given below, with $n = 6$:

In [2]:
```
H = Bool[0 1 0 0 0; 1 0 1 0 0; 0 1 1 0 0; 1 1 1 0 1; 0 1 0 1 1; 1 0 0 0 0
V = Bool[1 0 1 1 1 0; 0 1 0 0 1 1; 0 0 0 0 1 0; 0 1 0 1 0 0; 0 0 1 0 1 0]
```

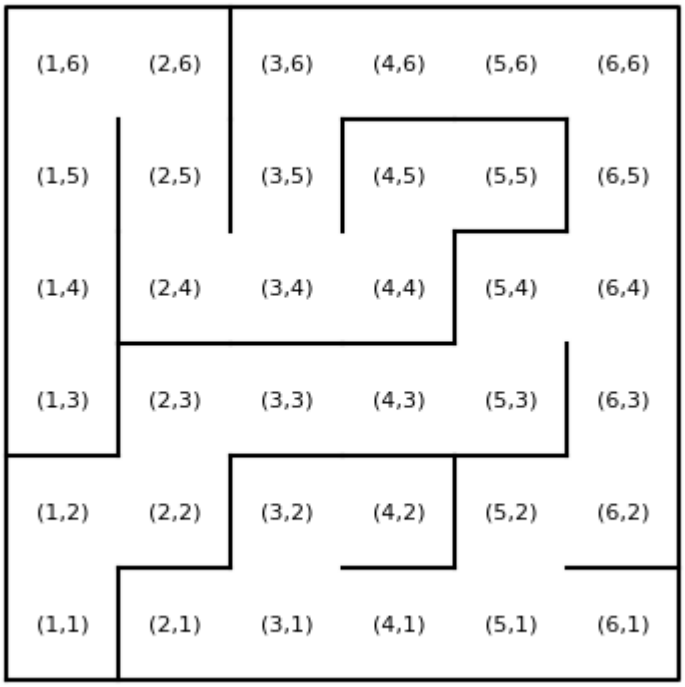
and the following helper functions can be used to plot the maze:

In [3]:
```
using PyPlot, Random

function plot_maze(H,V)
    clf()
    axis("off")
    axis("equal")
    n = size(H,1)
    plot([0,n,n,0,0], [0,0,n,n,0], color="k")

    for x = 1:n-1, y = 1:n
        if V[x,y]
            plot([x,x], [y-1,y], color="k")
        end
    end
    for x = 1:n, y = 1:n-1
        if H[x,y]
            plot([x-1,x], [y,y], color="k")
        end
```

```
    end
end

function plot_cell_indices(n)
    for i = 1:n
        for j = 1:n
            text(i-0.5, j-0.5, "($i,$j)",
                horizontalalignment="center",
                verticalalignment="center",
                fontsize=8)
        end
    end
end
```

Out[3]: plot_cell_indices (generic function with 1 method)

In [4]:
```
plot_maze(H,V)
plot_cell_indices(size(H,1))
```

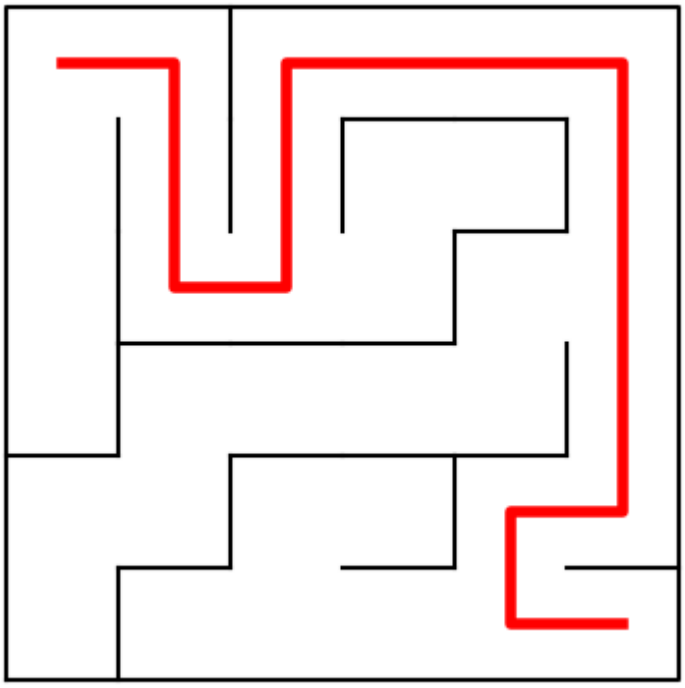


In addition, we will find paths between the points `1,n` and `n,1`, which can be stored in two arrays of integers. For the example above, this path is given by

In [5]:
```
x = [6, 5, 5, 6, 6, 6, 6, 6, 5, 4, 3, 3, 3, 2, 2, 2, 1];
y = [1, 1, 2, 2, 3, 4, 5, 6, 6, 6, 6, 5, 4, 4, 5, 6, 6];
```

and it can be plotted along with the maze using the commands:

In [6]:
```
plot_maze(H,V);
plot(x .- 0.5, y .- 0.5, color="r", linewidth=4);
```



Problem 1 - Generate random maze

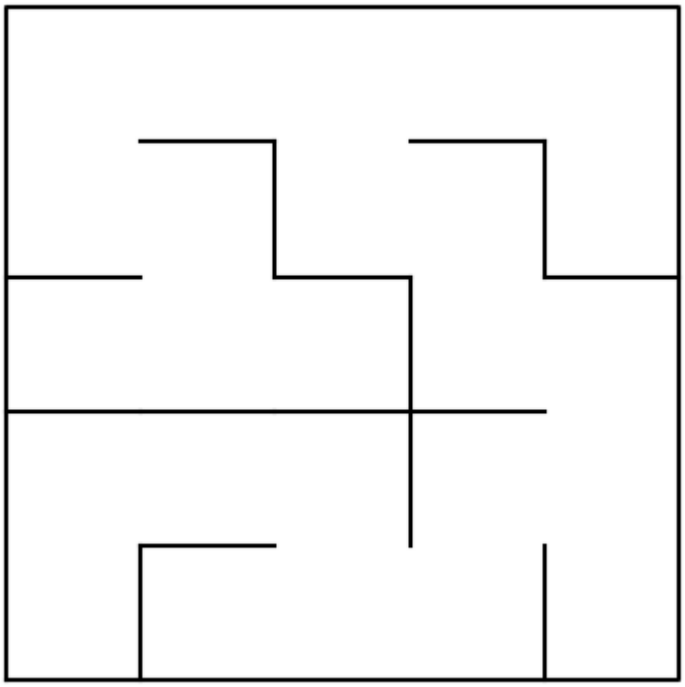
Write a function with the syntax

`H,V = make_maze(n)`
which produces a random maze of size `n` -by- `n` using the following algorithm:

1. Initialize `H` and `V` to matrices of `trues` (that is, assume all cells have walls on all sides)
2. Also initialize an array `visit` to a matrix of `falses`, to keep track of cells that have been visited
3. Create a function `dig(x,y)` which loops over the four directions (Right, Left, Up, Down) in a random order. For each direction, if the neighbor cell is valid and not visited, remove the corresponding wall from `H` or `V` and run the `dig` function recursively on the neighbor cell.
4. Call `dig(1,1)` and return `H,V`

In [7]:
```
function make_maze(n)
    H = trues(n, n-1)
    V = trues(n-1, n)
    visit = falses(n, n)
    function dig(x, y)
        visit[x, y] = true
        next_options = [(x, y-1), (x-1, y), (x+1, y), (x, y+1)]
        for next in randperm(length(next_options))
            next = next_options[next]
            if 1<=next[1]<=n && 1<=next[2]<=n && !visit[next[1], next[2]]
                if x != next[1]
                    V[min(x, next[1]), y] = false
                else
                    H[x, min(y, next[2])] = false
                end
```

```
                dig(next[1], next[2])
            end
        end
    end
    dig(1, 1)
    return H, V
end
H,V = make_maze(5)
plot_maze(H, V)
```



Problem 2 - Find path from `1,n` to `n,1`

Next, write a function with the syntax

`pathx, pathy = find_path(H,V)`
which finds a path in the maze `H,V` between the coordinates `1,n` and `n,1` using the following algorithm:

1. Again create an array `visit` to keep track of visited cells
2. Also initialize empty vectors `pathx,pathy` to store the final path
3. Create a recursive function `recur(x,y)` which performs the following:
   A. If the position `x==n` and `y==1` is found, insert these values into `pathx,pathy` and return `true`
   B. Otherwise, consider each neighbor of `x,y`. If the cell is valid, the maze has no wall in that direction, and the cell has not been visited, apply `recur` to the neighbor cell.
   C. If any of the calls to `recur` returns `true`, insert `x,y` into `pathx,pathy` and return `true`
4. Call `recur(1,n)` and return `pathx,path`

In [44]:
```
function find_path(H,V)
    n = size(H, 1)
    visit = falses(n, n)
    pathx = []
    pathy = []
    function neighbors(x, y)
        n_arr = []
        if x > 1 && !V[x-1, y]
            push!(n_arr, (x-1, y))
        end
        if x < n && !V[x, y]
            push!(n_arr, (x+1, y))
        end
        if y > 1 && !H[x, y-1]
            push!(n_arr, (x, y-1))
        end
        if y < n && !H[x, y]
            push!(n_arr, (x, y+1))
        end
        return n_arr
    end
    function recur(x, y)
        if x == n && y == 1
            push!(pathx, x)
            push!(pathy, y)
            return true
        end
        flag = false
        for next in neighbors(x, y)
            if !visit[next[1], next[2]]
                visit[next[1], next[2]] = true
                flag = flag || recur(next[1], next[2])
            end
        end
        if flag
            push!(pathx, x)
            push!(pathy, y)
        end
        return flag
    end
    recur(1, n)
    return pathx, pathy
end
```

Out[44]:
```
find_path (generic function with 1 method)
```

Problem 3 - Large maze test

Finally, run the code below to illustrate your codes.

In [45]:
```
n = 25
H,V = make_maze(n)
plot_maze(H,V)
x, y = find_path(H,Vz
print(x, y)
plot(x .- 0.5, y .- 0.5, color="r", linewidth=4);
```

```
Any[25, 25, 24, 24, 25, 25, 25, 25, 25, 24, 23, 22, 22, 21, 20, 19, 19, 2
0, 20, 21, 22, 22, 23, 23, 23, 22, 22, 21, 20, 19, 19, 18, 18, 17, 17, 18,
19, 19, 18, 17, 16, 15, 15, 16, 16, 16, 15, 14, 14, 14, 13, 13, 14, 14, 1
4, 13, 12, 12, 11, 10, 9, 9, 10, 11, 11, 11, 11, 12, 12, 12, 13, 14, 14, 1
5, 15, 15, 16, 17, 18, 18, 17, 17, 17, 16, 16, 15, 14, 14, 15, 15, 14, 14,
13, 13, 12, 11, 10, 9, 9, 10, 11, 12, 12, 11, 10, 9, 8, 8, 7, 7, 8, 8, 9,
10, 11, 12, 12, 11, 10, 10, 9, 9, 9, 8, 8, 7, 7, 7, 6, 6, 6, 6, 7, 7, 8,
8, 8, 7, 7, 7, 7, 7, 6, 6, 6, 6, 5, 5, 5, 4, 4, 3, 2, 2, 1]Any[1, 2, 2, 3,
3, 4, 5, 6, 7, 7, 7, 7, 6, 6, 6, 6, 5, 5, 4, 4, 4, 3, 3, 2, 1, 1, 2, 2, 2,
2, 1, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 3, 3, 2, 1, 1, 1, 2, 3, 3, 4, 4, 5,
6, 6, 6, 5, 5, 5, 5, 6, 6, 6, 7, 8, 9, 9, 8, 7, 7, 7, 8, 8, 7, 6, 6, 6, 6,
7, 7, 8, 9, 9, 10, 10, 10, 11, 11, 12, 12, 13, 13, 12, 12, 12, 12, 12, 11,
11, 11, 11, 10, 10, 10, 10, 10, 11, 11, 12, 12, 13, 13, 13, 13, 13, 14, 1
4, 14, 15, 15, 16, 17, 17, 16, 16, 15, 14, 14, 15, 16, 17, 17, 18, 18, 19,
20, 20, 21, 22, 23, 24, 24, 23, 22, 21, 21, 22, 23, 23, 24, 24, 24, 25, 2
5]
```