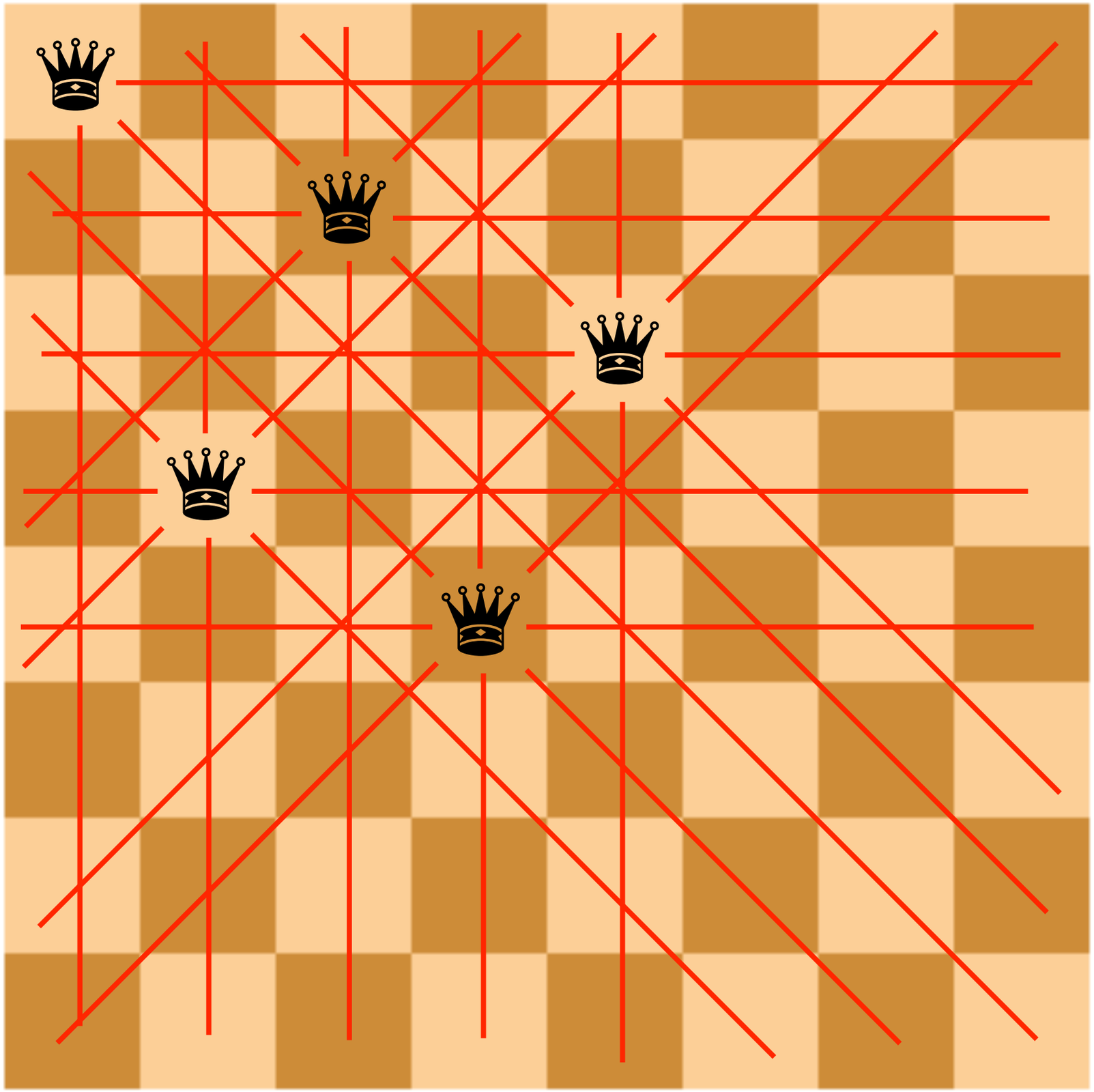
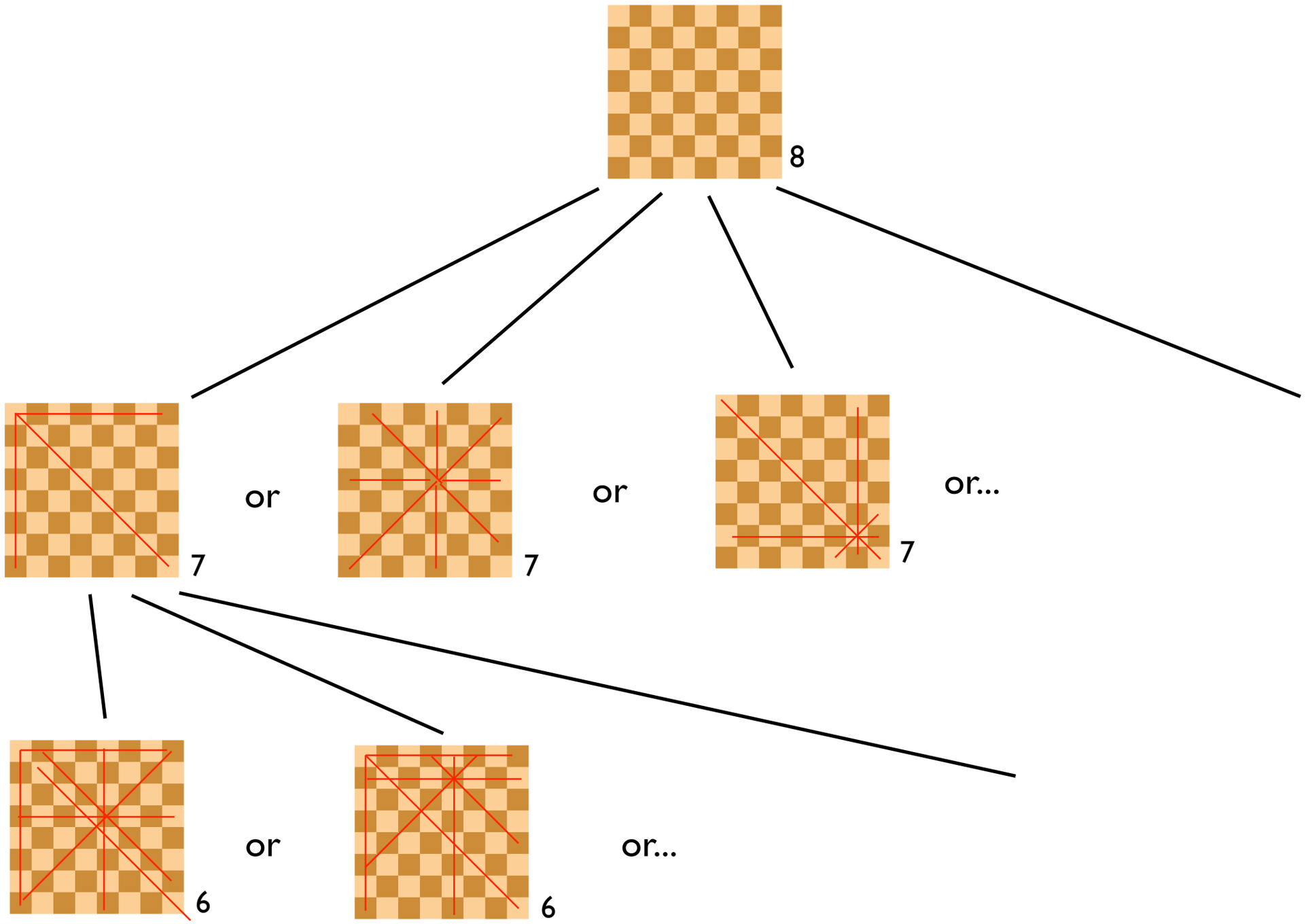
Recursive Backtracking, Round 2

(And recursive backtracking with no "backtracking")

Remember this?



Recursive Backtracking



The pattern

You have some *state*.

Chessboard
Sudoku board
Boggle board
(and non-board things)

The pattern

You have some *state*.

Chessboard
Sudoku board
Boggle board
(and non-board things)

If you're in a winning state, hooray!

Base Case

For each action you can take:

- Take that action.
- Recurse.
- See what happened.

Place a queen
Pick a number
Try a direction
(your problem here)

mazeEscape

```
   0 1 2 3 4
0  X X . X X
1  X . . . .
2  X X X X X
```

Can you escape from (r, c) in this maze in *exactly* n steps?

```
public boolean mazeEscape(char[][] map, int row, int col, int steps) {
```

mazeEscape

```
   0 1 2 3 4
0  X X . X X
1  X . . . .
2  X X X X X
```

Can you escape from (r, c) in this maze in *exactly* n steps?

(1, 1, 3) => true
(1, 1, 4) => true
(1, 1, _) => false

```
public boolean mazeEscape(char[][] map, int row, int col, int steps) {
```

Hint: your state is all four of these parameters!

A variation

1

```
  0 1 2 3 4
0 X X . X X
1 X . . . .
2 X X X X X
```

2

```
  0 1 2 3 4
0 X X . X X
1 X X . . .
2 X X X X X
```

3

```
  0 1 2 3 4
0 X X . X X
1 X X X . .
2 X X X X X
```

4

```
  0 1 2 3 4
0 X X . X X
1 X X . . .
2 X X X X X
```

A variation

Original Recursive Backtracking:

- Check your base case
- For each move:
  - Copy the world
  - Modify the copy
  - Recurse
  - See what happened

New Recursive Backtracking:

- Check your base case
- For each move:
  - Modify the world
  - Recurse
  - See what happened
  - If you didn't win, *undo the modification!*

I always forget to do this part.

What if you don't backtrack?

```
   0 1 2 3 4
0  X . . X X
1  . . . X .
2  . X X X .
3  . X . . .
4  . X X X X
5  . . . . .
```

Remind you of an APT? This should remind you of an APT...

What if you don't backtrack?

```
   0 1 2 3 4
0  X . . X X
1  . . . X .
2  . X X X .
3  . X . . .
4  . X X X X
5  . . . . .
```

Replace every '.' on the board with a number. Every 1 should only be adjacent to other 1s, each 2 only adjacent to other 2s, and so on.

Remind you of an APT? This should remind you of an APT...

What if you don't backtrack?

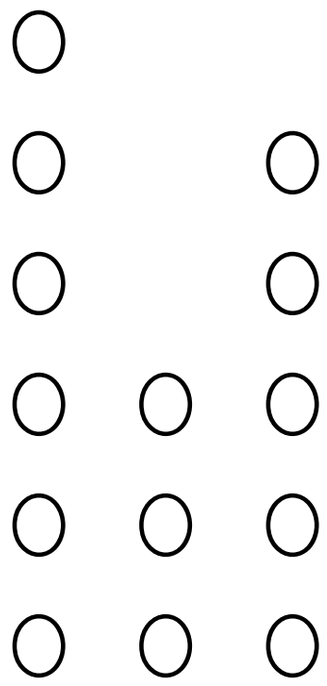
```
   0 1 2 3 4
0  X 1 1 X X
1  1 1 1 X 2
2  1 X X X 2
3  1 X 2 2 2
4  1 X X X X
5  1 1 1 1 1
```

Replace every '.' on the board with a number. Every 1 should only be adjacent to other 1s, each 2 only adjacent to other 2s, and so on.

```
public void reachability(char[][] map) {
```

Remind you of an APT? This should remind you of an APT...

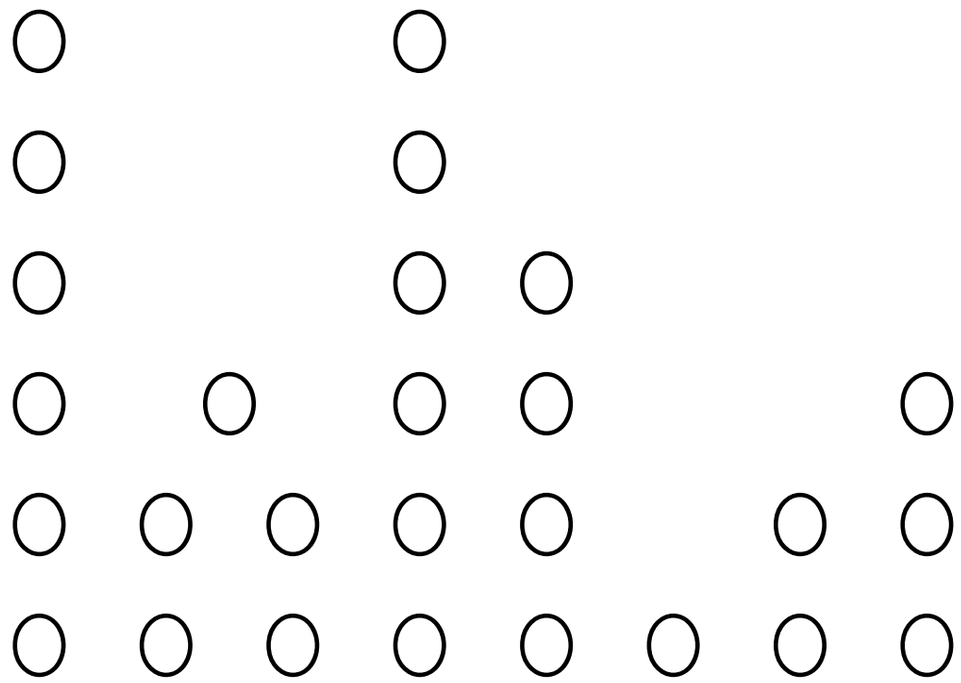
Multi-Heap Nim



```
public void multiHeapNim(ArrayList<Integer> piles) {
```

Remind you of an APT? This should remind you of an APT...

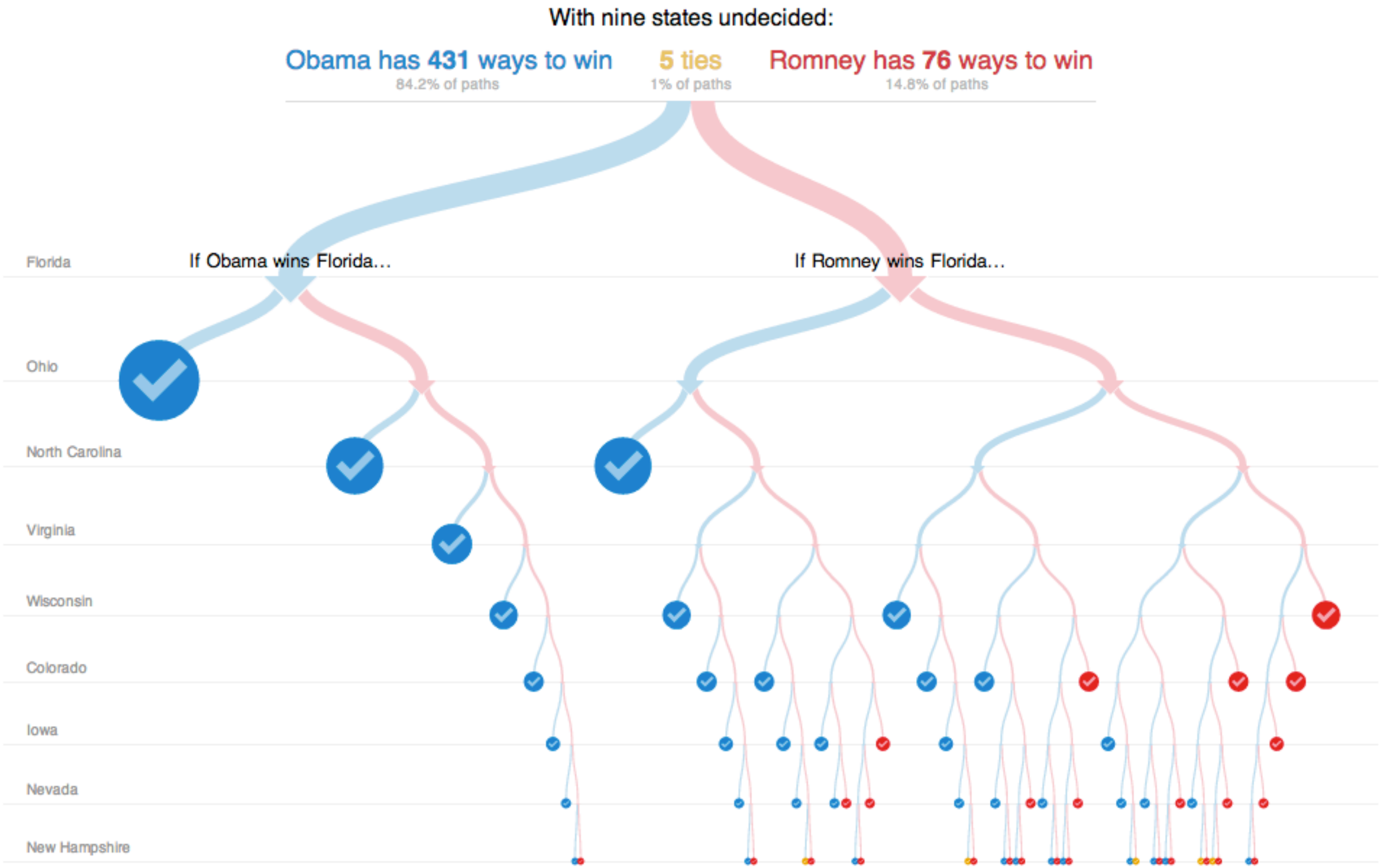
Multi-Heap Nim



```
public void multiHeapNim(ArrayList<Integer> piles) {
```

Remind you of an APT? This should remind you of an APT...

Democracy is a binary tree!



http://www.nytimes.com/interactive/2012/11/02/us/politics/paths-to-the-white-house.html