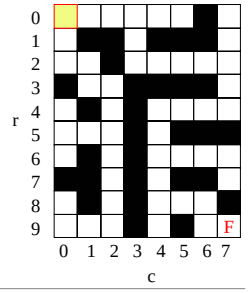
CSE 142

An Example of Recursion

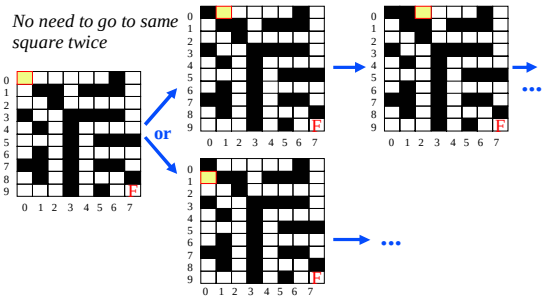
Path Planning

- Idea: want to discover if there is a path from square at 0,0 to square labeled F (which could be anywhere)
- Black squares represent obstacles
- Unless a path is blocked, can move up, down, left, or right



Redefine a hard problem as several simpler ones

No need to go to same square twice

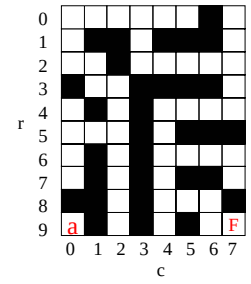


Simple Cases

- Define a class to represent maze objects
- State

```
char[ ][ ] maze;   // maze squares;
                   // 'X' in a square if
                   // square is visited or
                   // is an obstacle
```

- Base cases
  - If maze[r][c]=='F'
    Then "yes!"
  - If no place to go
    Then "no!"



Helper Method

- Check whether a proposed move is possible

```
/** Return true if it is possible to move to row r, column c, otherwise return false */
boolean legalMove (int r, int c) {
    return (r >= 0 && r < maze.length     &&
            c >=0 && c < maze[r].length &&
            maze[r][c] != 'X');
}
```

- Notice that this relies on && not evaluating its right operand if the left operand is false

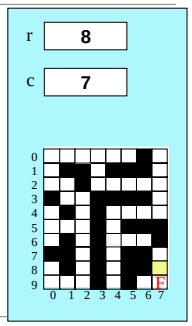
Elegant Solution

```
/** Return true if there is a path from <r,c> to an element of  the maze
 *  containing 'F', otherwise return false */
boolean isPath(int r, int c) {
    if (maze[r][c] == 'F') {
        return true;
    } else {
        maze[r][c] = 'X';
        return(  (legalMove(r-1,c)  && isPath(r-1,c))  ||
                 (legalMove(r+1,c) && isPath(r+1,c)) ||
                 (legalMove(r,c-1)  && isPath(r,c-1))  ||
                 (legalMove(r,c+1) && isPath(r,c+1)))
    }
}
```

Example

isPath(8, 7)

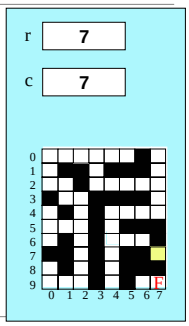
```
boolean isPath(int r, int c) {
    if (maze[r][c] == 'F') {
      return true;
    } else {
      maze[r][c] = 'X';
      return((legalMove(r-1,c)  && isPath(r-1,c))  ||
             (legalMove(r+1,c) && isPath(r+1,c)) ||
             (legalMove(r,c-1)  && isPath(r,c-1))  ||
             (legalMove(r,c+1) && isPath(r,c+1)))
    }
}
```



Example Cont

isPath(7, 7)

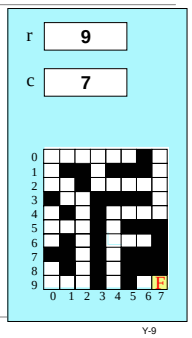
```
boolean isPath(int r, int c) {
    if (maze[r][c] == 'F') {
      return true;
    } else {
      maze[r][c] = 'X';
      return((legalMove(r-1,c)  && isPath(r-1,c))  ||
             (legalMove(r+1,c) && isPath(r+1,c)) ||
             (legalMove(r,c-1)  && isPath(r,c-1))  ||
             (legalMove(r,c+1) && isPath(r,c+1)))
    }
}
```



Example Cont

isPath(9, 7)

```
boolean isPath(int r, int c) {
    if (maze[r][c] == 'F') {
      return true;
    } else {
      maze[r][c] = 'X';
      return((legalMove(r-1,c)  && isPath(r-1,c))  ||
             (legalMove(r+1,c) && isPath(r+1,c)) ||
             (legalMove(r,c-1)  && isPath(r,c-1))  ||
             (legalMove(r,c+1) && isPath(r,c+1)))
    }
}
```



Recursion Summary

- A powerful programming technique & way of thinking
- Leads to elegant solutions of many problems
- Implementation in Java works because of the way method calls and local variable allocation already work