```
    private void iterateSearch(Dimension loc, int depth)
```

The class variable $isSearching$ is used to halt search, avoiding more solutions, once one path to the goal is found.

```
        if (isSearching == false) return;
```

We set the maze value to the depth for display purposes only:

```
        maze.setValue(loc.width, loc.height, (short)depth);
```

Here, we use the super class $getPossibleMoves$ method to get an array of possible neighboring squares that we could move to; we then loop over the four possible moves (a null value in the array indicates an illegal move):

```
Dimension [] moves = getPossibleMoves(loc);
for (int i=0; i<4; i++) {
if (moves[i] == null) break; // out of possible moves
                             // from this location
```

Record the next move in the search path array and check to see if we are done:

```
  searchPath[depth] = moves[i];
  if (equals(moves[i], goalLoc)) {
     System.out.println("Found the goal at " +
                        moves[i].width +
                        ``, " + moves[i].height);
     isSearching = false;
     maxDepth = depth;
     return;
  } else {
```

If the next possible move is not the goal move, we recursively call the iterateSearch method again, but starting from this new location and increasing the depth counter by one:

```
    iterateSearch(moves[i], depth + 1);
    if (isSearching == false) return;
  }
```