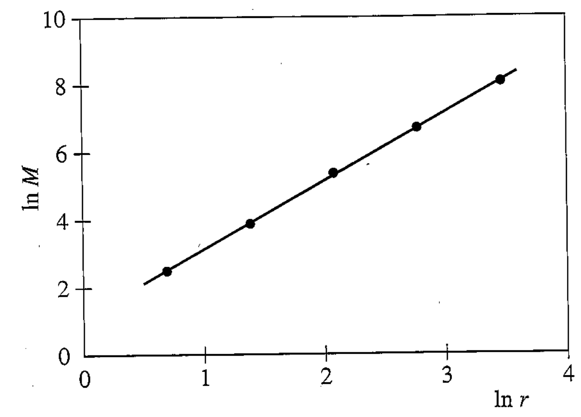

Figure 13.8 Plot of ln M versus ln r for a single Eden cluster generated on a $L = 61$ square lattice. A least squares fit from $r = 2$ to $r = 32$ yields a slope of approximately 2.01.

(b) Modify your program so that only the perimeter or growth sites are shown. Where are the majority of the perimeter sites relative to the center of the cluster? Grow as big a cluster as time permits. ■

Invasion percolation. A dynamical process known as *invasion percolation* has been used to model the shape of the oil-water interface that occurs when water is forced into a porous medium containing oil. The goal is to use the water to recover as much oil as possible. In this process a water cluster grows into the oil through the path of least resistance. Consider a lattice of size $L_x \times L_y$ with the water (the invader) initially occupying the left edge (see Figure 13.9). The resistance to the invader is given by assigning to each lattice site a uniformly distributed random number between 0 and 1; these numbers are fixed throughout the invasion. Sites that are nearest neighbors of the invader sites are the perimeter sites. At each time step, the perimeter site with the lowest random number is occupied by the invader and the oil (the defender) is displaced. The invading cluster grows until a path of occupied sites connects the left and right edges of the lattice. After this path forms, there is no need for the water to occupy any additional sites. To minimize boundary effects, periodic boundary conditions are used for the top and bottom edges, and all quantities are computed only over a central region far from the left and right edges of the lattice.

Class `Invasion` implements the invasion percolation algorithm. The two-dimensional array element `site[i][j]` initially stores a random number for the site at `(i,j)`. If the site at `(i,j)` is occupied, then `site[i][j]` is set equal to 1. If the site at `(i,j)` is a perimeter site, then `site[i][j]` is increased by 2. In this way we know which sites are perimeter sites, and the value of the random number associated with the perimeter site can still be determined. A new perimeter site is inserted into its proper ordered position in the lists `perimeterListX` and `perimeterListY`. The perimeter lists are ordered so that the site with the largest random number is at the beginning.

Two search methods are provided for determining the position of a new perimeter site in the perimeter lists. In a *linear search* we go through the list in order until the random number associated with the new perimeter site is between two random numbers in the list. In a *binary search* we divide the list in two and determine in which half the new random

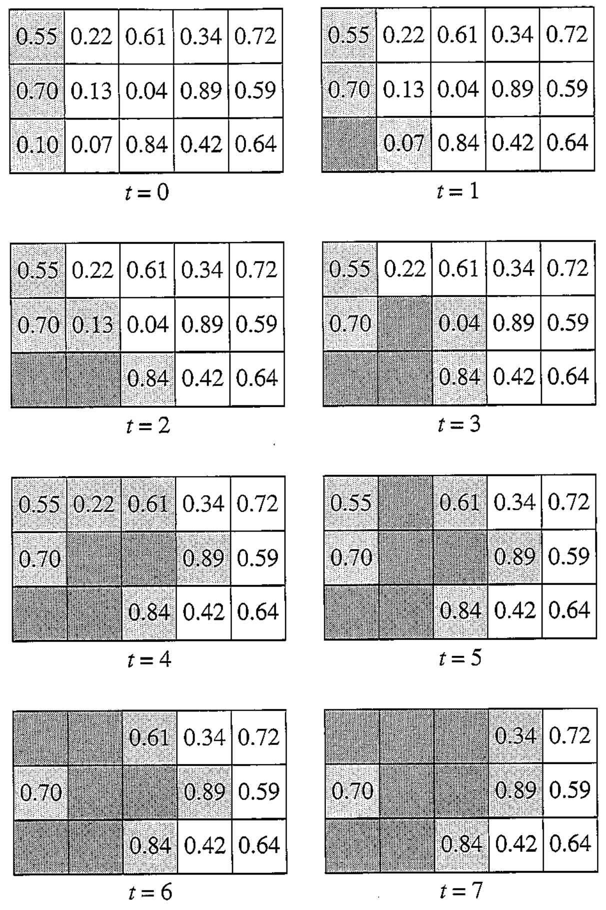

Figure 13.9 Example of a cluster formed by invasion percolation on a 5×3 lattice. The lattice at $t = 0$ shows the random numbers that have been assigned to the sites. The darkly shaded sites are occupied by the invader that occupies the perimeter site (lightly shaded) with the smallest random number. The cluster continues to grow until a site in the right-most column is occupied.

number belongs. Then we divide this half into half again and so on until the correct position is found. The linear and binary search methods are compared in Problem 13.7d. The binary search is the default method used in class `Invasion`.

The main quantities of interest are the fraction of sites occupied by the invader and the probability $P(r)\Delta r$ that a site with a random number between r and $r + \Delta r$ is occupied. The properties of invasion percolation are explored in Problem 13.7.

Listing 13.4 Class for simulating invasion percolation.

```
package org.opensourcephysics.sip.ch13.invasion;
import java.awt.Color;
import org.opensourcephysics.frames.*;

public class Invasion {
   public int Lx, Ly;
   public double site[][];
```