

## 11.2 Spreading of Fire

### *Mathematica* Quick Review Questions

*Introduction to Computational Science: Modeling and Simulation for the Sciences*

Angela B. Shiflet and George W. Shiflet

Wofford College

© 2006 by Princeton University Press

This file contains system-dependent Quick Review Questions and answers in *Mathematica* for Module 11.2 on "Spreading of Fire." Complete all code development in *Mathematica*.

### Initializing the System

**Quick Review Question 1** Suppose the *fire* function begins as follows:

```
fire[n_, probTree_, probBurning_, chanceLightning_, chanceImmune_, t_]
```

Part a assigns values to global variables, and Parts b and c initialize *forest* to be an  $n$ -by- $n$  matrix (table) and *grids* to be a list containing *forest*. Quick Review Question 9 completes the implementation of the function *fire*.

- a. Each site is assigned a value as described in the algorithm for Cell Initialization. Complete the code to make *forest* and *forestExtended* local variables and to assign values to global variables *probLightning* and *probImmune*.

```
:= _____[_____forest, forestExtended_____,
```

```
(** Assign values to global variables **)  
probLightning = chanceLightning;  
probImmune = chanceImmune;
```

- b. Complete the code to initialize *forest* to be an  $n$ -by- $n$  matrix (table). Each cell is assigned a value as described in the algorithm for Cell Initialization.

```
forest = _____[  
  If[_____, _____ probTree,  
    If[Random[] < probBurning, _____, _____],  
  _____],  
  {n}, _____]
```

- c. Give command to establish *grids* as a list containing *forest*. At the end of the simulation, *grids* holds all the simulation grids.

### Updating Rules

**Quick Review Question 2** The following questions develop the rule for *spread[site, N, E, S, W]* that applies to the situation where a site does not contain a tree at this or any time step:

- a. Select the value of *site*: *EMPTY, TREE, BURNING, \_*, none of these
- b. The values at the neighboring sites—*N, E, S*, and *W*—are irrelevant. Select the other parameters: *EMPTY, TREE, BURNING, \_*, none of these
- c. Select the return value: *EMPTY, TREE, BURNING, \_*, none of these
- d. Write the entire rule.

**Quick Review Question 3** The following questions develop the rule for *spread[site, N, E, S, W]* that applies to the situation where a site contains a burning tree:

- a. Select the value of *site*: *EMPTY, TREE, BURNING, \_*, none of these
- b. Because the values at the neighboring sites—*N, E, S*, and *W*—are irrelevant, we can use the blank pattern for each of the next four parameters. Give the blank pattern.
- c. Because a burning tree always burns down, give the return value of the *spread* function for this situation.
- d. Write the entire rule.

**Quick Review Question 4** The following questions develop the rule for *spread[site, N, E, S, W]* that applies to the situation where a site contains a non-burning tree that may catch fire because a neighboring site contains a burning tree:

- a. Select the value of *site*: *EMPTY, TREE, BURNING, \_*, none of these
- b. Select the meaning of the following call to *If*:

`If[Random[] < probImmune, TREE, BURNING]`

- A. If a random number is less than the probability of immunity, then the tree catches fire; else it does not.
  - B. If a random number is less than the probability of immunity, then the tree does not catch fire; else it does.
  - C. If a random number is less than the probability of immunity, then the tree stays immune; else it does not.
  - D. If a random number is less than the probability of immunity, then the tree does not stay immune; else it does.
- c. For the tree to have a chance of burning due to fire at a neighboring site, give the value that at least one of *N, E, S, W* must have.
  - d. Give the call to *MatchQ* to test if one of the parameters *N, E, S, W* is *BURNING*.
  - e. Select the operator to activate the definition if *MatchQ* test returns *True*:
 

A. /	B. /.	C. /;
D. /:	E. \	F. \.
G. \;	H. \:	I. none
  - f. Give the entire rule.

**Quick Review Question 5** Complete the following rule for  $spread[site, N, E, S, W]$  that applies to the situation where a site contains a non-burning tree that may be hit by lightning and burn:

```
spread[TREE, N_, E_, S_, W_] :=
  _____[_____ < probLightning * (1 - probImmune), _____, _____]
```

The following segment contains all the updating rules for the function  $spread$ :

```
(* Empty cell remains empty cell next time step *)
spread[EMPTY, _, _, _, _] := EMPTY;

(* Burning tree results in empty cell next time step *)
spread[BURNING, _, _, _, _] := EMPTY;

(* Perhaps next time step tree with burning neighbor(s) burns itself *)
spread[TREE, N_, E_, S_, W_] :=
  If[Random[] < probImmune, TREE, BURNING] /;
  MatchQ[BURNING, N | E | S | W];

(* Perhaps tree is hit by lightning and burns next time step *)
spread[TREE, N_, E_, S_, W_] :=
  If[Random[] < probLightning*(1 - probImmune), BURNING, TREE];
```

### Periodic Boundary Conditions

**Quick Review Question 6** This question extends a matrix as in Figure 11.2.6 by attaching the last row to the beginning and the first row to the end of the original matrix.

- Write a call to the  $Take$  function to return a list containing the last row of matrix  $mat$ .
- Write a call to a function to return a list containing the first row of matrix  $mat$ .
- Complete the following statement to make  $matNS$  an extended matrix of  $mat$  as described in this question.

```
matNS = _____[Take[mat, _____], _____, Take[mat, _____]]
```

**Quick Review Question 7** This question extends a matrix as in Figure 11.2.7.

- Write a statement to make  $trans$  a matrix with the rows and columns of matrix  $matNS$  switched.
- Make  $matEW$  a matrix containing the concatenation of the last row of  $trans$ ,  $trans$ , and the first row of  $trans$ . Use  $Take$ .
- Give a call to a function to return a matrix with the rows and columns of  $matEW$  switched.
- If the original matrix  $mat$  is of size 7-by-7, after extending the matrix as in this and the previous Quick Review Question, give the size of the extended matrix.

### Applying a Function to Each Grid Point

**Quick Review Question 8** This question develops the function *applyExtended*.

- a. Complete the code to start the definition of *applyExtended*, which is to have parameters *fnc* and *matExt* and local variables *n*, *site*, *N*, *E*, *S*, and *W*.

```
applyExtended[fnc_, matExt_] := _____[___n, site, N, E, S, W]...
```

- b. Select the symbol that separates immediately after } in Part a:  
 A. ,            B. .            C. ;            D. :  
 E. \_            F. /.           G. /;           H. /:
- c. Select the symbol from Part b that separates the statements in the body of the function *applyExtended*.
- d. Write the statement to assign to *n* the length of the internal (non-extended) matrix.
- e. We can use the function *Table* to generate the return matrix. Within the body of *Table*, we assign values to *site*, *N*, *E*, *S*, and *W*, and apply *fnc* with these parameters for each internal cell site. Select the symbol from Part b that separates the statements in the body of *Table*.
- f. Suppose *i* represents the row index and *j* the column index. To apply the function *fnc* to each internal cell of *matExt*, we let *i* (and *j*) vary between two values. Give the initial value of *i* (or *j*).
- g. Give the final value of *i* (or *j*).
- h. Select the symbol from Part b that separates the body of *Table* from the range of the indices,  $\{i, 2, n + 1\}$ ,  $\{j, 2, n + 1\}$ .
- i. Figure 11.2.9 gives the coordinates of a site and its neighbors. Give the code to assign to *site* the value of the (*i*, *j*)-element of matrix *matExt*.
- j. Give the code to assign to *N* the value from *matExt* corresponding to the neighbor to the north.
- k. Give the code to assign to *E* the value from *matExt* corresponding to the neighbor to the east.
- l. Give the complete definition of *applyExtended*.

### Simulation Program

**Quick Review Question 9** Implement the loop in the *fire* function, assuming *grids* is a list containing the initial forest Quick Review Question 1 develops.

### Display Simulation

**Quick Review Question 10** This question develops the function *showGraphs* that produces a graphic corresponding to each simulation matrix in a list (*graphList*).

- a. Give the function call to return the number of elements in *graphList*.
- b. Give the statement to assign to local variable *g* the *k*-th element in *graphList*.
- c. Complete the command to produce a graphic of *g* as a rectangular grid with yellow for *EMPTY*, green for *TREE*, and burnt orange for *BURNING*.

```
Show[Graphics[_____ [Reverse[g]] _____
  {EMPTY -> RGBColor[1, 1, 0],
   TREE -> RGBColor[0.1, 0.75, 0.2],
   BURNING -> RGBColor[0.6, 0.2, 0.1]}],
 AspectRatio -> Automatic]
```

- d. Give the entire definition of *showGraphs*.

### Answers to Quick Review Question

1. a. 

```
:= Module[{forest, forestExtended},
  (** Assign values to global variables **)
  probLightning = chanceLightning;
  probImmune = chanceImmune;
```

  - b. 

```
forest = Table[
  If[Random[] < probTree,
    If[Random[] < probBurning, BURNING, TREE],
    EMPTY],
  {n}, {n}]
```
  - c. 

```
grids = {forest};
```
2. a. *EMPTY*  
b. *\_*  
c. *EMPTY*  
d. 

```
spread[EMPTY, _, _, _, _] := EMPTY
```
3. a. *BURNING*  
b. *\_*, the underscore character  
c. *EMPTY*, which indicates an empty cell  
d. 

```
spread[BURNING, _, _, _, _] := EMPTY
```
4. a. *TREE*  
b. B. If a random number is less than the probability of immunity, then the tree does not catch fire; else it does.  
c. *BURNING (2)*  
d. 

```
MatchQ[BURNING, N | E | S | W]
```

  
e. C. *!*;  
f. 

```
spread[TREE, N_, E_, S_, W_] :=
  If[Random[] < probImmune, TREE, BURNING]
  /; MatchQ[BURNING, N | E | S | W];
```
5. 

```
spread[TREE, N_, E_, S_, W_] :=
  If[Random[] < probLightning * (1 - probImmune), BURNING, TREE]
```
6. a. 

```
Take[mat, -1]
```

  
b. 

```
Take[mat, 1]
```

  
c. 

```
matNS = Join[Take[mat, -1], mat, Take[mat, 1]]
```

7. a. `trans = Transpose[matNS]`  
 b. `transeW = Join[Take[trans, -1], trans, Take[trans, 1]]`  
 c. `Transpose[transeW]`  
 d. 9-by-9
8. a. `applyExtended[fnc_, matExt_] := Module[{n, site, N, E, S, W}...`  
 b. A. ,  
 c. C. ;  
 d. `n = Length[matExt] - 2;`  
 e. C. ;  
 f. 2  
 g. `n + 1`  
 h. A. ,  
 i. `site = matExt[[i]][[j]]` OR `site = matExt[[i, j]]`  
 j. `N = matExt[[i - 1]][[j]]` OR `N = matExt[[i - 1, j]]`  
 k. `E = matExt[[i]][[j + 1]]` OR `E = matExt[[i, j + 1]]`  
 l. `applyExtended[fnc_, matExt_] := Module[{n, site, N, E, S, W},`  
     `n = Length[matExt] - 2;`  
     Table[  
         `site = matExt[[i]][[j]];`  
         `N = matExt[[i - 1]][[j]];`  
         `E = matExt[[i]][[j + 1]];`  
         `S = matExt[[i + 1]][[j]];`  
         `W = matExt[[i]][[j - 1]];`  
         `fnc[site, N, E, S, W],`  
         {`i, 2, n + 1`}, {`j, 2, n + 1`}}  
     ]  
 ];
9. Do[  
     `forestExtended = extendMat1[forest];`  
     `forest = applyExtend[spread, forestExtended];`  
     `grids = Append[grids, forest],`  
     {t}  
 ];
10. a. `Length[graphList]`  
 b. `g = graphList[[k]]`  
 c. `Show[Graphics[RasterArray[Reverse[g]] /.`  
     {`EMPTY` -> `RGBColor[1, 1, 0]`,  
     `TREE` -> `RGBColor[0.1, 0.75, 0.2]`,  
     `BURNING` -> `RGBColor[0.6, 0.2, 0.1]}`},  
     `AspectRatio -> Automatic]`

```
d. showGraphs[graphList_] := Module[{k, g},
  Do[
    g = graphList[[k]];    (* grid at k-th time step *)

    Show[Graphics[RasterArray[Reverse[g]] /.
      {EMPTY -> RGBColor[1, 1, 0],
      TREE -> RGBColor[0.1, 0.75, 0.2],
      BURNING -> RGBColor[0.6, 0.2, 0.1]}],
      AspectRatio -> Automatic,
      {k, Length[graphList]}
    ]
  ]
]
```