11.3 Movement of Ants

Maple Quick Review Questions

Introduction to Computational Science: Modeling and Simulation for the Sciences
Angela B. Shiflet and George W. Shiflet
Wofford College

This file contains system-dependent Quick Review Questions and answers in *Maple* for Module 11.3 on "Movement of Ants." Complete all code development in *Maple*.

Grid Initialization

Quick Review Question 1 This question refers to the initialization of the grid for ant movement. Suppose we establish constants *EMPTY*, *NORTH*, *EAST*, *SOUTH*, and *WEST* as representative characters, as follows:

```
EMPTY:= "T":
NORTH :="N":
EAST := "E":
SOUTH := "S":
WEST := "W":
```

a. Write a statement to assign to *EMPTY* the character "T" without displaying the result. Similar statements give character values "N", "E", "S", and "W" to *NORTH*, *EAST*, *SOUTH*, and *WEST*, respectively.

b. Write a function call to return a random integer between 1 and 4.

c. We define a list, *dir*, of directions, as follows:

```
dir := [NORTH, EAST, SOUTH, WEST]:
```

Suppose *r* is an integer between 1 and 4. Give the expression to return the direction from *dir* with index *r*.

d. Complete the code to assign to *initialGrid* an *n*-by-*n* array of ordered pairs. The first member of each ordered pair is zero. With a probability of *probAnt*, the cell contains an ant that faces in a random direction. Otherwise, the cell does not contain an ant. Assume that *rand0to1* is a function that returns a random floating point number between 0 and 1.

```
initialGrid:=
    [_______________([_______________(
      [_____________, `if`(_________ < _________,
         dir[rand(1..4)()], __________)],
     j = 1..n)], i = 1..n)]:
```

e. Complete the loop to generate a chemical trail with no ants on the middle row of the grid *initialGrid*. The maximum amount of chemical, 50, occurs in column *n*, and for column *j* the amount of chemical is a fraction, *j* / *n*, expressed as an integer, of the maximum 50. For example, if *j* is 10 and *n* is 17, then the amount of chemical is the integer 29 because (50)(10)/17 = 29.41.

```
j from 1 to n do
    grid[[IntegerPart[n/2] + 1, __________] := [__________, __________]
end do:
```

Sensing

Quick Review Question 2 With integer parameter *a* representing the amount of chemical, complete the code for the first *sense* rule that an empty cell does not sense.

```
define(sense,
   sense[_____________, b::list, c::list, d::list, e::list) =
      _____________);
```

Quick Review Question 3 This question refers to the second *sense* rule that an ant turns at random in the direction of one of the neighboring cells with the greatest amount of chemical. We begin by defining a function, *senseMax*, that returns the direction, *NORTH*, *EAST*, *SOUTH*, *WEST*, of the maximum of four arguments, respectively. If more than one argument is equal to the maximum, a direction is picked at random. Assume the function definition begins as follows:

```
senseMax := proc(n, e, s, w)
   local mx, mxDirList, lng;
…
```

a. Complete the statement to assign the maximum level of chemical in neighboring cells to *mx*.

```
mx := _____________:
```

b. Complete the statement to assign to *mxDirList* a list of directions (*NORTH*, *EAST*, *SOUTH*, *WEST*) where the maximum level, *mx*, occurs. If a parameter is not equal to *mx*, the component in the resulting list should be *NULL*, indicating no value.

```
mxDirList := [_____________ (_____________, NORTH, NULL),
              _____________ (_____________, EAST,  NULL),
              _____________ (_____________, SOUTH, NULL),
              _____________ (_____________, WEST,  NULL)]:
```

c. Complete the statement to assign to *lng* the number of elements in *mxDirList*.

```
lng := _____________(mxDirList):
```

d. Complete the statement to return a random position from 1 and *lng*, inclusively, in *mxDirList*.

```
mxDirList[__________________];
```

e. Complete the additional rule for *sense* (the first rule is in Quick Review Question 2), where each of the five parameters is a list consisting of an integer and a character parameter. The rule uses delayed evaluation of *senseMax*.

```
definemore(sense,
   sense([a____________, aa________________],
         [b____________, nn________________],
         [e____________, ee________________],
         [s____________, ss________________],
         [w____________, ww________________]) =
   [________________, _____________(b, e, s, w)]
);
```

Walking without Concern for Collision

Quick Review Question 4 Complete the definition of the first *walk* rule: For a cell that remains empty, the amount of chemical decrements by one but does not fall below 0. Because *Maple* applies the first rule that matches, as we see shortly, this rule is not the first we define.

```
_____________(walk,
   walk([a_____________, _____________], N::list, E::list,
      S::list, W::list, NE::list, SE::list,
      SW::list, NW::list, Nn::list, Ee::list,
      Ss::list, Ww::list) = [_____________, _____________]
);
```

Quick Review Question 5 Complete the definition of one form of the second *walk* bullet: If an ant wants to go north and the cell to the north is empty, the ant leaves the current cell and the amount of chemical at that site increments by one. Similar rules apply to the east, south, and west directions.

```
definemore(walk,
   walk([a_____________, _____________], [b_____________, ___________],
      E::list, S::list, W::list,
      NE::list, SE::list, SW::list, NW::list,
      Nn::list, Ee::list,
      Ss::list, Ww::list) = [_____________, _____________],
…
```

Quick Review Question 6 Complete the definition of the third *walk* rule: If an ant stays in a cell, the amount of chemical remains the same.

```
definemore(walk,
   walk([a_____________, c_____________], N::list,
```

```
      E::list, S::list, W::list,
      NE::list, SE::list, SW::list, NW::list,
      Nn::list, Ee::list,
      Ss::list, Ww::list) = [_____________, _____________]
);
```

Quick Review Question 7

a. Complete the definition of one form of the fourth *walk* bullet: If an ant moves to a cell, the cell continues to have its same amount of chemical. In this case, the current site is initially empty and the cell to the north contains an ant that is facing south. Similar rules apply to ants in the east, south, and west directions facing the current site.

```
definemore(walk,
   walk([a::integer, _____________], [b::integer, _____________],
      E::list, S::list, W::list,
      NE::list, SE::list, SW::list, NW::list,
      Nn::list, Ee::list,
      Ss::list, Ww::list) = [_____________, _____________],
   …
```

b. The pattern of arguments for this rule matches a rule for which previous Quick Review Question?

Walking with Concern for Collision

Quick Review Question 8

a. Complete the definition of the *walk* rule for the situation in Figure 11.3.1 in which a collision should be avoided for the current ant that faces an empty cell to the north and a northeast ant that faces west. Because this rule is more specific than that of some previous Quick Review Questions, we define it first.

```
_____________(walk,
   walk([a::integer, _____________], [b::integer, _____________],
      E::list, S::list, W::list,
      [ne::integer, _____________], SE::list, SW::list,
      NW::list, Nn::list,
      Ee::list, Ss::list, Ww::list) =
         [_____________, _____________],
…
```

b. Give the number of similar such rules.

c. The pattern of arguments for this rule matches a rule for which previous Quick Review Questions?

Quick Review Question 9

a. Complete the definition of the *walk* rule for the situation in Figure 11.3.1 in which a collision should be avoided where the current site is empty but ants to

the north and east both want to move into the site. In this situation, if the amount of chemical at the site is positive, it decrements by 1.

```
definemore(walk,
   walk([a::integer, _____________], [b::integer, _____________],
      [e::integer,_____________], S::list, W::list,
      NE::list, SE::list, SW::list, NW::list,
      Nn::list, Ee::list, Ss::list, Ww::list)=
      [max(a - 1, 0), _____________],
…
```

b. Give the number of similar such rules.

c. The pattern of arguments for this rule matches a rule for which previous Quick Review Questions?

Visualizing the Simulation

Quick Review Question 10 Suppose *graphList* is a list of square grids representing frames of the simulation at consecutive time steps; and *maxChem* is the maximum amount of chemical in any cell of any grid in *graphList*. The function *showGraphs* uses a function *matchColor* to associate the value of a cell with a color. The definition of the latter begins as follows:

```
matchColor := proc(mat, i, j)
   local siteColorAmt:
…
```

a. Write a statement to indicate that variable *maxChem* is not local to *matchColor*.

b. Complete the statement to assign to *siteColorAmt* the amount of chemical component, the second coordinate of the ordered pair *mat*[*i*, *j*], scaled to be between 0.5 and 1.0, as described in this section.

```
siteColorAmt := _____________ - (_____________] * 0.5)/_____________:
```

c. Complete the *if* statement to return an RGB color designation for a cell. If the cell is empty, return a shade of gray (equal amounts of red, green, and blue) based on *siteColorAmt*. Otherwise, return a shade of red based on *siteColorAmt*.

```
if (mat[i, j, _____________] = _____________) then
   _____________ (_____________, _____________, _____________, _____________);
else
   _____________ (_____________, _____________, _____________, _____________);
end if;
```

The complete function is as follows:

```
matchColor := proc(mat, i, j)
   local siteColorAmt:
   global maxChem:

   siteColorAmt := 1.0 - (mat[i, j, 1] * 0.5)/maxChem:
```

```
   if (mat[i, j, 2] = EMPTY) then
      COLOR(RGB, siteColorAmt, siteColorAmt, siteColorAmt);
   else
      COLOR(RGB, siteColorAmt, 0, 0);
   end if;
end proc:
```

d. The definition *showGraphs* to visualize the grids in *graphList* with maximum chemical amount of *maxChem* begins as follows:

```
showGraphs := proc(graphList, maxChem)
   local k, g, aSquare, plotGrid, plotList, n;
…
```

Write a segment to indicate that *maxAmtChem* is not local and to give it the value of parameter *maxChem*.

e. Write a statement to assign to *g* the *k*th grid, or square matrix, in *graphList*.

f. Write a statement to assign to *aSquare* a list containing the vertices of a unit square in the first quadrant with a vertex at the origin. Start with the origin and list the vertices in counterclockwise order.

g. Complete the statement to assign to *plotGrid* a sequence of squares, colored as designated by *matchColor* and translated into the rows and columns corresponding to the grid *g*.

```
plotGrid := _____________(_____________ (
            plottools[_____________](
               plots[_____________](aSquare,
               axes = none,
               scaling = CONSTRAINED,
               color = _____________),
            _______, ______),
         j = 1..n), i = 1..n):
```

h. Complete the statement to append to *plotList* a display of all the squares in the sequence from Part g

```
plotList := [__________(plotList), ___________ [__________](plotGrid)]:
```

i. Complete the statement to generate an animation of the elements in *plotList*.

```
___________ [__________](plotList, ___________ = _________);
```

Answers to Quick Review Questions

1. a. `EMPTY := "T":`

The assignments for the other directions are as follows:

```
NORTH := "N":
EAST  := "E":
SOUTH := "S":
WEST  := "W":
```

b. `rand(1..4)()`

c. `dir[r]`

Thus, the following expression returns a random direction, *N*, *E*, *S*, or *W*:

```
dir[rand(1..4)()]
```

d.
```
initialGrid :=
   [seq([seq([0, `if`(rand0to1() < probAnt,
               dir[rand(1..4)()], EMPTY)],
         j = 1..n)], i = 1..n)]:
```

e.
```
for j from 1 to n do
   initialGrid[trunc(n/2) + 1, j] :=
         [trunc(maxChem*j/n), EMPTY]
end do:
```

2.
```
define(sense,
     sense([a::integer, EMPTY], b::list, c::list, d::list, e::list) =
           [a, EMPTY]);
```

3. **a.**
```
mx := max(n, e, s, w):
```

b.
```
mxDirList := [`if`(n = mx, NORTH, NULL),
              `if`(e = mx, EAST,  NULL),
              `if`(s = mx, SOUTH, NULL),
              `if`(w = mx, WEST,  NULL):]:
```

c.
```
lng = nops(mxDirList):
```

d.
```
mxDirList[rand(1..lng)()];
```

The complete definition of is as follows:

```
senseMax := proc(n, e, s, w)
   local mx, mxDirList, lng;
   mx := max(n, e, s, w):
   mxDirList := [`if`(n = mx, NORTH, NULL),
                 `if`(e = mx, EAST,  NULL),
                 `if`(s = mx, SOUTH, NULL),
                 `if`(w = mx, WEST,  NULL)]:
   lng := nops(mxDirList):
   mxDirList[rand(1..lng)()];
end proc:
```

e.
```
definemore(sense,
   sense([a::integer, aa::character],
         [b::integer, nn::character],
         [e::integer, ee::character],
         [s::integer, ss::character],
         [w::integer, ww::character]) =
   [a, 'senseMax'(b, e, s, w)]
);
```

4.
```
definemore(walk,
   walk([a::integer, EMPTY], N::list, E::list,
      S::list, W::list, NE::list, SE::list,
      SW::list, NW::list, Nn::list, Ee::list,
      Ss::list, Ww::list) = [max(a - 1, 0), EMPTY]
);
```

5.
```
definemore(walk,
     walk([a::integer, NORTH], [b::integer, EMPTY],
           E::list, S::list, W::list,
```

```
            NE::list, SE::list, SW::list, NW::list,
            Nn::list, Ee::list,
            Ss::list, Ww::list) = [`if`(a>0, a+1, 0), EMPTY],
        …
```

6.
```
definemore(walk,
     walk([a::integer, c::character], N::list,
          E::list, S::list, W::list,
          NE::list, SE::list, SW::list, NW::list,
          Nn::list, Ee::list,
          Ss::list, Ww::list) = [a, c]
     );
```

7. **a.**
```
definemore(walk,
   walk([a::integer, EMPTY], [b::integer, SOUTH],
      E::list, S::list, W::list,
      NE::list, SE::list, SW::list, NW::list,
      Nn::list, Ee::list,
      Ss::list, Ww::list) = [a, SOUTH],
```

b. Quick Review Question 4. Thus, for the rule of the current, more specific situation ever to apply, its definition must appear before the rule of Quick Review Question 4.

8. **a.**
```
define(walk,
   walk([a::integer, NORTH], [b::integer, EMPTY],
      E::list, S::list, W::list,
      [ne::integer, WEST], SE::list, SW::list,
      NW::list, Nn::list,
      Ee::list, Ss::list, Ww::list) =
         [a, NORTH],
   …
```

b. 11

c. Quick Review Questions 5 and 6. Thus, for the rule of the current, more specific situation ever to apply, its definition must appear before the rules of Quick Review Questions 5 and 6.

9. **a.**
```
definemore(walk,
   walk([a::integer, EMPTY], [b::integer, SOUTH],
      [e::integer,WEST], S::list, W::list,
      NE::list, SE::list, SW::list, NW::list,
      Nn::list, Ee::list, Ss::list, Ww::list)=
      [max(a - 1, 0), EMPTY],
   …
```

b. 5

c. Quick Review Questions 4, 6, and 7. Thus, for the rule of the current, more specific situation ever to apply, its definition must appear before the rules of Quick Review Questions 4, 6, and 7.

10. **a.**
```
global maxChem:
```

b.
```
siteColorAmt := 1.0 - (mat[i, j, 1] * 0.5)/maxChem:
```

c.
```
if (mat[i, j, 2] = EMPTY) then
   COLOR(RGB, siteColorAmt, siteColorAmt, siteColorAmt);
else
   COLOR(RGB, siteColorAmt, 0, 0);
end if;
```

d.
```
global maxAmtChem:
maxAmtChem := maxChem:
```

e.
```
global maxAmtChem:
maxAmtChem := maxChem:
```

f.
```
g := graphList[k]:
```

g.
```
aSquare := [[0, 0], [0, 1], [1, 1], [1, 0]]:
```

g.
```
plotGrid := seq(seq(
            plottools[translate](
               plots[polygonplot](aSquare,
               axes = none,
               scaling = CONSTRAINED,
               color = matchColor(g, i, j)),
            j, i),
         j = 1..n), i = 1..n):
```

h.
```
plotList := [op(plotList), plots[display](plotGrid)]:
```

i.
```
plots[display](plotList, insequence = true);
```