**A step-by-step example of how the VBA program AnalyzePatch works**

Suppose we have the following landscape to analyze:



After defining variables and clearing out old edge and core count data, the next step of the program is to copy this landscape, which is in the Raw\_Range, into the Plot\_Range (cells C10–L19). The program does this as a “Paste special”—that is, it pastes the *value* of each cell, not the formulae (in case you used formulae to generate the raw data). As you can see from the code SkipBlanks:=False, the program does not skip blanks as it copies cells, so it clears any previous data that may be in the Plot\_Range. Therefore, before we enter the outer loop, the Plot\_Range looks exactly like the Raw\_Range:



Before entering the outer loop, the program sets the variable Patch\_Count to 0 and the variable No\_Remaining\_Forested\_Cells to False. The outer loop will increase Patch\_Count by 1 each time it loops, and it will stop running when No\_Remaining\_Forested\_Cells is set to True.

The outer loop performs only three functions before the program moves to the inner loop: it increases the variable Patch\_Count by 1, it sets the variable No\_New\_Neighbors to False, and it sets the variable First\_Cell\_This\_Patch to False. The inner loop will repeat until No\_New\_Neighbors is set to True. The first step of the inner loop is to set the variable New\_Neighbor\_Found to False.

Up to this point, nothing has happened to the cells that were copied into Plot\_Range. Now we enter the part of the program defined by the commands For Each Plot\_Cell in Range(Plot\_Range) . . . Next Plot\_Cell. This part of the program takes a look at every cell in Plot\_Range, starting at the upper left (cell C10). When the program comes to the first If . . . End If section, it checks to see if C10 contains the letter f *and* if the variable First\_Cell\_This\_Patch is False. The cell does not contain the letter f, so even though First\_Cell\_This\_Patch is False, the program skips down to the End If statement and continues to the next If . . . End If section.

The second If . . . End If section also checks to see if C10 contains the letter f. Since C10 is empty, the program again skips down to the following End If statement and moves on to the next command. This is the Next Plot\_Cell statement, which tells the program to repeat the process on the next cell in Plot\_Range, cell D10.

Unlike cell C10, cell D10 contains the letter f, and when we get to the first If . . . End If section, we find that *both* conditions of the If statement are satisfied (the cell contains the letter f *and* First\_Cell\_This\_Patch is set to False), so the program proceeds to the commands following the If statement. This section first changes the value of cell D10 to 1 (the current value of Patch\_Count). Then it changes First\_Cell\_This\_Patch to True so that the program will skip this section until a new patch is started, and it changes New\_Neighbor\_Found to True so that the program will continue looking for other neighbors. The program then checks the second If . . . End If section, but cell D10 no longer contains the letter f (it was just changed to the number 1), so it skips this section again.

The Next Plot\_Cell command continues to tell the program to check each cell in the Plot\_Range, but all of these are empty up to cell D14, the next cell that contains the letter f, so the program skips over both If . . . End If sections each time. When the program reaches cell D14, it still skips the first If . . . End If section (because First\_Cell\_This\_Patch is True). Cell D14 meets the first condition of the second If . . . End If section—it contains the letter f—but it fails the second condition, which is that at least one of the four cells adjacent to D14 must contain the current value of Patch\_Count. In fact, none of the remaining cells will meet both of these conditions; the program will check them all, but do nothing with any of them.

Once the program has checked every Plot\_Cell in the Plot\_Range in order, it moves beyond the Next Plot\_Cell statement for the first time. Here it encounters the statement If New\_Neighbor\_Found = False Then No\_New\_Neighbors = True. The current value of New\_Neighbor\_Found is True, however, so the program leaves the variable No\_New\_Neighbors alone (it is set to False at this point). It may be obvious to us at this time that there are no new cells to add to the first habitat patch, but we will see in the next loop that the program won’t always identify all the cells in a patch on its first pass through the Plot\_Range, so as long as it has found one new neighbor, it will continue running the inner loop.

After the first pass through the inner loop, the Plot\_Range looks like Figure xxx. The value of No\_New\_Neighbors is still False, so the program sets New\_Neighbor\_Found to False and checks every cell in the Plot\_Range again. This time through, it still skips the first If . . . End If section because First\_Cell\_This\_Patch is still True, and it skips the second If . . . End If section because none of the cells that contain the letter f are adjacent to a cell that contains the number 1. At the end of this pass, New\_Neighbor\_Found is still False, so No\_New\_Neighbors is set to True, and the program finally leaves the inner loop.



Figure xxx

Back in the outer loop, the program checks to see if First\_Cell\_This\_Patch is False. This variable was set to True back in the first If . . . End If section of the inner loop, so the program does *not* change No\_Remaining\_Forested\_Cells to True. The following Loop statement sends the program back to the beginning of the outer loop, and since No\_Remaining\_Forested\_Cells is still False, the outer loop is repeated.

Once again the outer loop adds 1 to Patch\_Count (its value becomes 2), sets No\_New\_Neighbors to False, and sets First\_Cell\_This\_Patch to False before entering the inner loop. The inner loop sets New\_Neighbor\_Found to False and then tells the program to start checking every cell again. This time, the program makes it all the way to cell D14 before it finds a cell that contains the letter f. Since First\_Cell\_This\_Patch has been set back to False, the program enters the first If . . . End If section, changes the f in cell D14 to the number 2 (the current value of Patch\_Count), and sets First\_Cell\_This\_Patch and New\_Neighbor\_Found to True. The Plot\_Range now looks like Figure xxx+1.



Figure xxx+1

The program searches for new cells that contain the letter f by working from left to right, then top to bottom, as if it were reading a book. Thus, as the program continues searching from cell D14, the next cell with the letter f that it encounters will be cell F14. The second If . . . End If section will check to see if F14 is adjacent to a cell with the number 2; since cell F15 has not yet been assigned a number, the program will skip to the End If statement and go on to investigate cell G14, without assigning the number 2 to cell F14. The program will not investigate cell F14 again on this pass through the inner loop. This is why the program needs to continue making passes through the entire Plot\_Range until New\_Neighbor\_Found is not set to True.

After F14, the next cell that contains the letter f will be D15. This time, when the second If . . . End If section checks the cells adjacent to D15 it will find that one of them (D14) contains the current Patch\_Count number. It will therefore set the value of D15 to 2 and set New\_Neighbor\_Found to True (though in this case, it is already True). The program will fail to recognize that F15 belongs to patch number 2, but it will catch all the others, and at the end of this pass through the inner loop, the Plot\_Range will look like Figure xxx+2.



Figure xxx+2

Since New\_Neighbor\_Found is True, the program will go back through the inner loop another time, setting New\_Neighbor\_Found to False again. This time it will *still* not recognize that cell F14 is part of patch 2, but it will find that cell F15 is next to a cell that contains the number 2 (cell F16), so it will change the value of F15 to 2 and set New\_Neighbor\_Found back to True, thereby keeping the program from exiting the inner loop. In the next pass through the loop, the program will finally see that F14 belongs in patch 2, but when it changes the value of F14 to 2 it will also set New\_Neighbor\_Found to True. The program will therefore have to make one more pass through the inner loop; as always, it sets New\_Neighbor\_Found to False at the beginning of the loop, but this time the variable remains False. Therefore, at the end of the loop, No\_New\_Neighbors is set to True, and we finally exit the inner loop. The Plot\_Range should now look like Figure xxx+3.



Figure xxx+3

At this point, it looks like we’re done, but the program still needs to get out of the outer loop. Unfortunately, First\_Cell\_This\_Patch is True (it was set to True in the inner loop when the program identified the first cell in patch 2), so No\_Remaining\_Forested\_Cells remains False at the end of the outer loop, and the program makes one more pass through both the outer and inner loops. As always, the outer loop adds one to Patch\_Count (moving it up to 3), and sets First\_Cell\_This\_Patch to False. When the inner loop runs this time, the program will find no cells that contain the letter f, so it won’t set First\_Cell\_This\_Patch to True, and it will exit the inner loop without changing anything. Therefore, at the end of the outer loop, First\_Cell\_This\_Patch will still be False, so No\_Remaining\_Forested\_Cells will be set to True, and the program will finally exit the outer loop. Note that because of this last pass through the outer loop, the variable Patch\_Count holds a number one larger than the actual number of patches in the landscape.