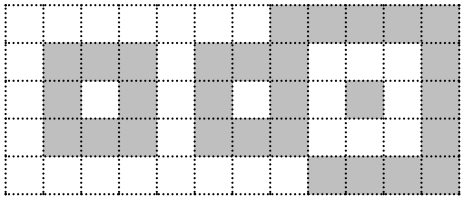
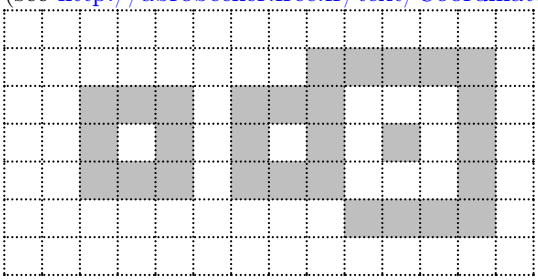
Finding ocean, islands, lakes, etc... in a board

Alain Brobecker

Your array is  → water (0) → ground (1)

Step 1: Surround it with the base value (here water)
(see http://abrobecker.free.fr/text/CoordinateSystemForHexagonalBoards.pdf to know why)



Step 2: Fill the outer zone with -1 by starting on (0;0) and using the **recursive fill routine** below, it will reach everything due to the surrounding made on step 1. Maybe set TestValue and FillValue as global variables?

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1						-1
-1	-1				-1								-1
-1	-1				-1								-1
-1	-1				-1								-1
-1	-1	-1	-1	-1	-1	-1	-1	-1					-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

```
//No need to make boundary tests because we added surroundings!
void Fill(int x,int y,char TestValue,char FillValue) {
   Board[x][y]=FillValue;
   if(Board[x-1][y]!=TestValue) { Fill(x-1,y,TestValue,FillValue); }
   if(Board[x][y-1]!=TestValue) { Fill(x,y-1,TestValue,FillValue); }
   if(Board[x+1][y]!=TestValue) { Fill(x+1,y,TestValue,FillValue); }
   if(Board[x][y+1]!=TestValue) { Fill(x,y+1,TestValue,FillValue); }
}
```

Step 3: As long as you can found a $\geqslant 0$ value near a < 0 value, fill it with -2 (same routine as above)

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-2	-2	-2	-2	-2	-1
-1	-1	-2	-2	-2	-1	-2	-2	-2				-2	-1
-1	-1	-2		-2	-1	-2		-2				-2	-1
-1	-1	-2	-2	-2	-1	-2	-2	-2				-2	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	-2	-2	-2	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Step 4: As long as you can found a $\geqslant 0$ value near a < 0 value, fill it with -3
Step 5: As long as you can found a $\geqslant 0$ value near a < 0 value, fill it with -4

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-2	-2	-2	-2	-2	-1
-1	-1	-2	-2	-2	-1	-2	-2	-2	-3	-3	-3	-2	-1
-1	-1	-2	-3	-2	-1	-2	-3	-2	-3	-4	-3	-2	-1
-1	-1	-2	-2	-2	-1	-2	-2	-2	-3	-3	-3	-2	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	-2	-2	-2	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Please note that if convert all values < -2 to -2, then it will give you the two islands without the lakes. If you convert all values < -3 to -3 it will give you all three lakes but without the inner island...
To process more than one island, you have to search entry point of first island, process it, clear it (again using the fill routine), then search entry point of second island, process it, clear it...

Alternative method:
Once you have filled your first zone with -2, you could immediatly search for a $\geqslant 0$ value near a -2 and set it to -3, then search for a $\geqslant 0$ value near a -3, etc...

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1						-1
-1	-1	-2	-2	-2	-1								-1
-1	-1	-2	-3	-2	-1								-1
-1	-1	-2	-2	-2	-1								-1
-1	-1	-1	-1	-1	-1	-1	-1	-1					-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

And when you can't go deeper, you go back in the recursion stack. Here it would be search next $\geqslant 0$ value near a -1 and set it to -4, then search for a $\geqslant 0$ value near a -4, then go back in the recursion stack, which will trigger a $\geqslant 0$ value near a -4...

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-4	-4	-4	-4	-4	-1
-1	-1	-2	-2	-2	-1	-4	-4	-4	-6	-6	-6	-4	-1
-1	-1	-2	-3	-2	-1	-4	-5	-4	-6	-7	-6	-4	-1
-1	-1	-2	-2	-2	-1	-4	-4	-4	-6	-6	-6	-4	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-4	-4	-4	-4	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

But you have more recursion (which is ok if you have control on the depth of your recursion stack and know the maximum recursion depth) and you will need to save multiple boundaries. Here $[-2;-3]$ is first island+lake, $[-4;-7]$ is second island with its two lakes and inner island, when $[-6;-7]$ is the lake with inner island.