Computer Graphics

Flood Fill
Homework 1 Results

Zoie

Heidi

Frances

Anaid

Yujun
Homework 1 Results

Katherine

Grace
Homework 1 Results

Claire

Rhys
function drawLine(x1, y1, x2, y2, color) {
  graphics.fillStyle = color;
  var dx = x2-x1;
  var dy = y2-y1;
  if (Math.abs(dx)>Math.abs(dy)) {
    if (x2<x1) {
      drawLine(x2, y2, x1, y1, color);
    } else {
      var m = dy/dx; // slope
      var b = (x2*y1-x1*y2)/dx; // y intercept
      for (x = Math.floor(x1); x <= Math.floor(x2); x++) {
        y = m*x+b;
        fillPixel(x,y);
      }
    }
  } else {
    if (y2<y1) {
      drawLine(x2, y2, x1, y1, color);
    } else {
      var w = dx/dy; // inverse slope
      var c = (y2*x1-y1*x2)/dy; // x intercept
      for (y = Math.floor(y1); y <= Math.floor(y2); y++) {
        x = w*y+c;
        fillPixel(x,y);
      }
    }
  }
}
HackSmith 2021
March 26th-27th
25:11:34:52 until hacking begins.
Register Now

https://hacksmithcollege.com/
Review

After watching the content video, you should be able to...

• Define the 2D fill operation
• Determine 4-connected and 8-connected regions
• Design a recursive function with stop condition & simplification
• Implement a recursive 2D fill algorithm
• Pseudocode & simulate a sweep-based 2D fill algorithm
• Explain the advantages of the sweep-based fill.
Q. I always feel like we learn recursion, and then almost never use it because there are generally better ways to use (memoization etc.), can you give a specific example where recursion is better than others? Or is memoization always better than recursion?

A. Implementing flood fill without recursion would be difficult, and would require an additional data structure (e.g., a stack)

Q. How should we decide if using recursion or iteration for a programming problem?

A. Convenience and efficiency. If the recursive leads to a more concise program, and doesn’t decrease efficiency, then you should use it.
Q. What are some other things besides recursion fill that recursion is useful for?

A. Recursion proves useful in a number of algorithms known as “divide-and-conquer” where a problem is repeatedly split in half for a rapid solution. Examples include Quicksort and the Fast Fourier Transform.

Q. How might one go about doing multiple colours with recursion?

A. I’m assuming you mean doing a gradient or texture fill. In this case, the procedure for figuring out which pixels get replaced is the same. Once we’ve decided to fill a pixel, we have to compute its proper color under whatever multicolored pattern we’re using, based on its (x,y) loc.
Q. I'm confused about slide 7 with the recursion example of 2018, I understand how the 2 is found and then printed, but don't really get how the rest are found. You explain it as going to the last step and then doing x%10, but wouldn't that give you 20%10 = 2 or am I thinking of that wrong?

A. Each recursive call has a different argument and thus produces a different digit.

Note that 20%20 = 0, not 2. 20/10 = 2.
Q. If you're making 80 recursive calls to only fill 20 pixels, is that inefficient/slow?

A. There is some overhead to each call. But, it is hard to avoid making 80 checks somehow, since any pixel could potentially have four neighbors that need coloring.

Sweep fill improves the situation somewhat (only 40 calls for 20 pixels)

Q. Not a question, but the in-depth example was super helpful!!!

A. Thanks!
Q. I'm not really sure what the first question is asking? After seeing the correct answer I think I can sort of piece it together but I thought it was asking about the stack overflow or how many calls need to be made to fill an object, which I thought was just 4?

A. The call depth is the number of calls made, minus the number of returns/backtracks. In flood fill, it’s sort of like the length of a string leading back to the start point.
Your Questions

Q. In the sweep fill operation example, can you explain why the row above row 1 (A-E) was not filled second? Why would the call be from pixel C instead of A or E, the first and last pixels of the row?

Q. For sweep fill, how do you know which pixel to go to after you reach a dead end? For example, how do you know you go from C to J in the video clip?
Other Questions?
Handout 2

Practice on:
• Thinking about recursion
• Recursive fill
• Sweep fill
Fill with limited recursive calls:

- North only
- North and East only
- North, East, and South only
Could the stack overflow problem be addressed by running four separate recursion operations, one for each quadrant? What problems, if any, do you see with this approach?
Handout 2