Lecture 9/5/2025 - Floodfill

Thursday, August 21, 2025 9:03 PM

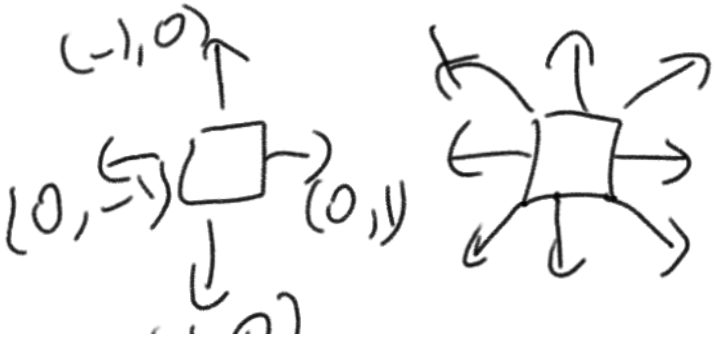
Microsoft Paint

Click Dot filled enclosed region w/same color

```
fill(int x, int y, grid)
    grid[x][y] = fillchar
// Don't do if already fill
// Don't fill neighbor if
// out of bounds, or boundary
// square
```

How do I handle iterating through all the neighbor squares?

Answer: DR/DC or DX/DY arrays.



(1,0)

```
int DR[4] = {-1, 0, 0, 1}; // DX
int DC[4] = {0, -1, 1, 0}; // DY
// at (r, c)
for (int i = 0; i < 4; i++) {
    nR = r + DR[i];
    nC = c + DC[i];
```

r 2
c 3

i	nR	nC	
0	2-1	3	(1,3)
1	2	3-1	(2,2)
2	2	3+1	(2,4)
3	2+1	3	(3,3)

for 8 directions

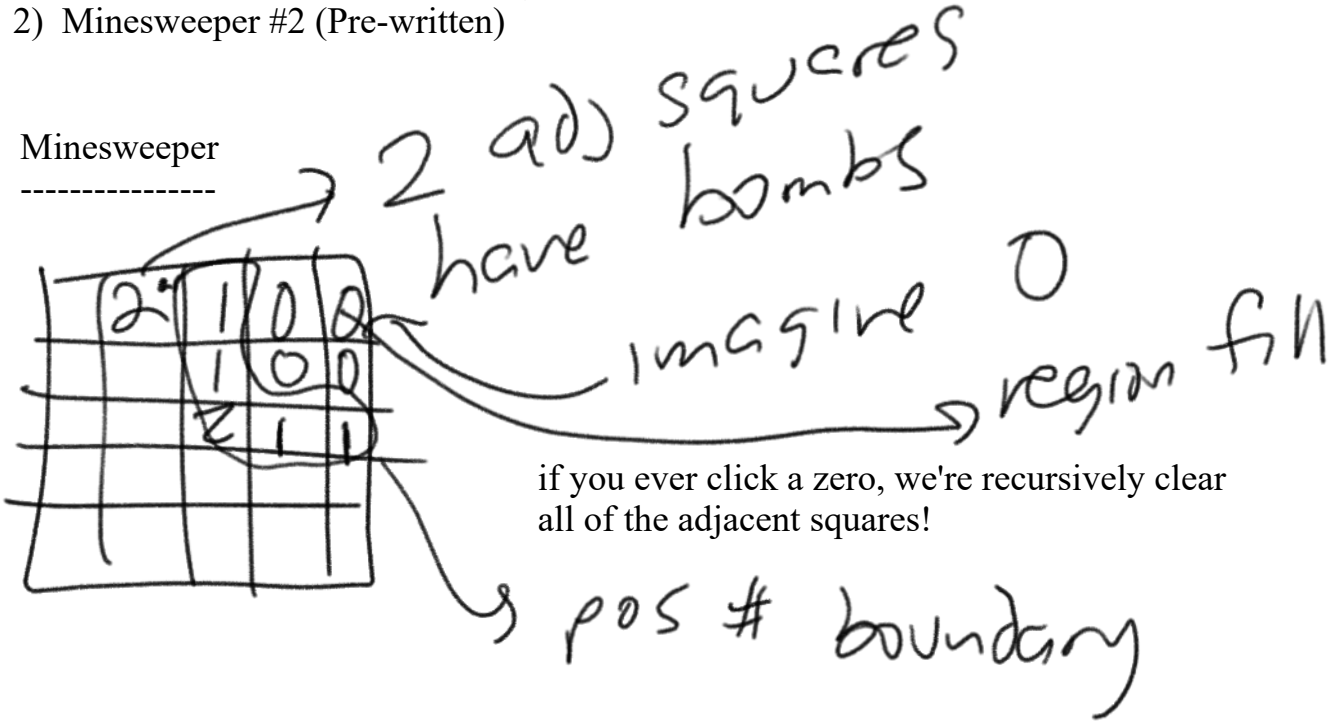
```
int DR[8] = {-1, -1, -1, 0, 0, 1, 1, 1};
int DC[8] = {-1, 0, 1, -1, 1, -1, 0, 1};
```

```
void fill(int r, int c, int** grid) {
```

```
        grid[r][c] = '*';
        for (int i=0; i<4; i++) {
            int nR = r + DR[i];
            int nC = c + DC[i];

            // Don't go out of bounds.
            if (!inbounds(nR, nC)) continue;

            // Already filled.
            if (grid[nR][nC] == '*') continue;

            // Blocked square or boundary of fill region.
            if (grid[nR][nC] == BOUNDARY) continue;

            fill(nR, nC)
        }
    }
```

Two applications
---
1) Kattis Problem Called Counting Stars
2) Minesweeper #2 (Pre-written)

Minesweeper



if you ever click a zero, we're recursively clear all of the adjacent squares!

for minesweeper we have

DX = {-1, -1, -1, 0, 0, 1, 1, 1};
DY = {-1, 0, 1, -1, 1, -1, 0 1}

click

DY = {-1, 0, 1, -1, 1, -1, 0 1}

getBombs