Decimal Spiral Explanation

Jonathan Lam
Tutor: Lucy Qiu

August 10, 2018

Solving the decimal spiral can be split up into multiple sections. In hindsight, I could have improved the code in various aspects (both style-wise and mathematically) but I've already spent 9 hours on an optional question worth no marks. so yeah nah.

1 Count the amount of numbers

I started off by counting the amount of 'numbers' i.e. not a dash, that were in the square grid. This was just a modified version of the 'spiral' exercise, where each `printf("#");` was replaced with a `countNum++` instead. The function `countNumbers(size)` takes the size as an input and returns the amount of numbers in the grid.

A natural consequence of this would be to calculate the first term (top left corner of the grid). Since the spiral starts with a zero in the centre, we just subtract one and to get the last digit, we mod 10 it.

2 First row

Doing the first row is easy. You just get the starting number and subtract one along the way (taking mod 10). This was implemented by saying `startingNum - col + 1` (to adjust for the indices).

3 Second row

The second row is also pretty easy. Put dashes along the whole entire row, except for the last column, in which case you subtract 1 from the last column from the first row.

```
void row2(int col, int startingNum, int size) {
    if (col == size) {
        printf("%d", (startingNum+100 - col%10)%10);
    } else {
        printf("-");
    }
}
```

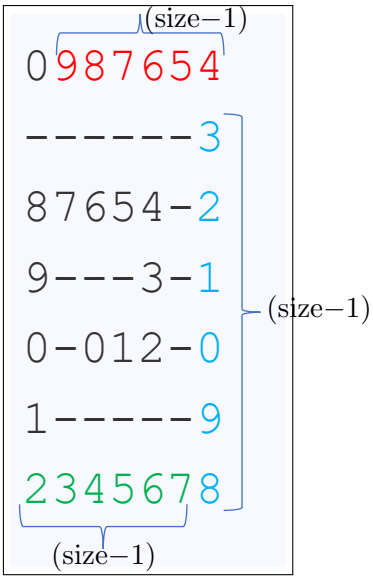
`lastColumnFirstRow = (startingNum - col + 1)` (from section 2). And if you subtract 1 from this, you get the lastColumnSecondRow, which is `startingNum - col`.

The reason I added 100 was to prevent the number going into the negatives (which we do not want). Also the reason why I wrote `col%10` was so that I'm always subtracting a number less than 10 (e.g. subtract 4 instead of 14) - again to ensure my number doesn't go into the negatives. Finally the whole expression is taken mod 10 to get the last digit.

4 Last Row

First we calculate the starting number for the last row, and then we add 1 along the row.
(i.e. `lastRowStartNum + col`)

To get the starting number for the last row, get the starting number from the first row and subtract $3 \times (\text{size} - 1) \pmod{10}$



So we can see that we start with 0 (the first number on the top row) and subtract the length of the reds (size−1) and then the blues (size−1) and then the greens (size−1). And this gives us the starting number for the last column, which is 2.

5 Second Last Row

The second last row has numbers in the first column and last column, and dashes everywhere else.

Hence the code is gonna look something like this:

```
void row2ndLast(int col, int lastRowStartNum, int size) {
    if (col == 1) {
        // print first number
    } else if (col == size) {
        // print last number
    } else {
        printf("-");
    }
}
```

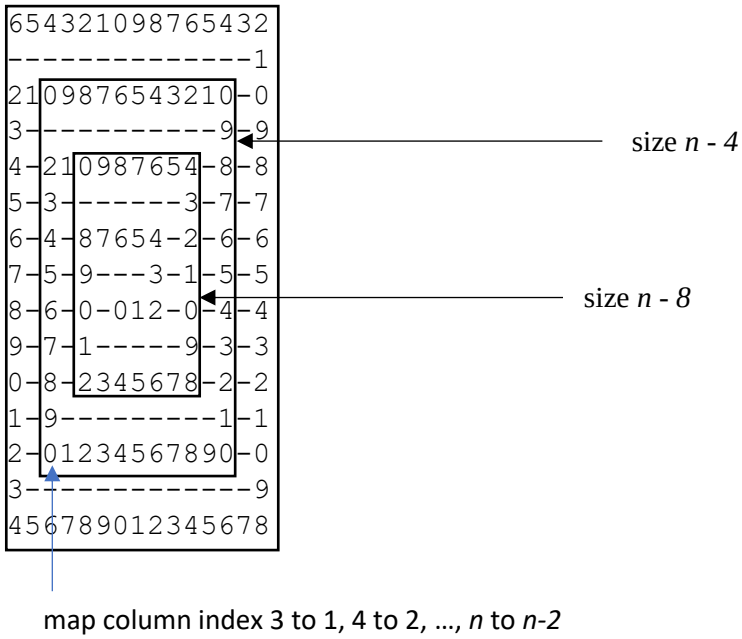
To calculate the last digit, this is `lastRowStartingNumber - 1`.

The last column in the second last row is lastRowStartNum + (size-1) + 1. We add `(size-1)` to get the last number on the last row, and then add 1 to get move up into the second last row.

This could have been simplified since $-1 + 1 = 0$ but I did not do this in my code since performance wasn't an issue, and it helped my readability (so I could see exactly where the formula came from).

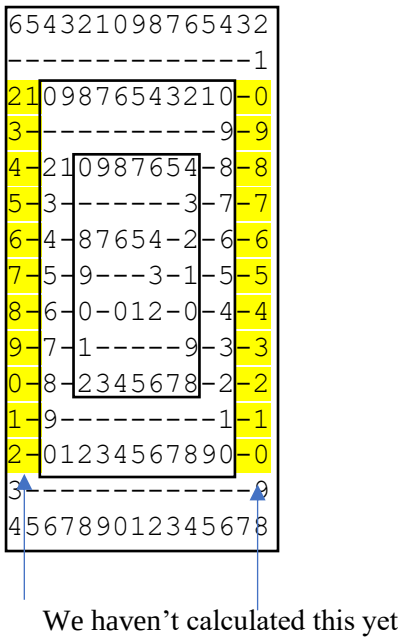
6 The middle

We note that we can truncate the outer layers of the grid to get a smaller grid inside. We can use recursion to calculate the inside grid. We just need to give parameters for the new size and new starting number. The size is now size − 4 since we removed two of the layers away on each side.



So for example, to calculate the middle box, this is simply the same code as generating a decimal_spiral of `size=11`, startingNum = 0 and column and row indices shifted by 2 (so column 1 in the small box now becomes column 3 in the big box). The use of this recursion allows you to calculate the insides of the boxes.

All that is left is to calculate the outer edges of the middle section i.e. the part in the yellow



Well this looks easy! The second column and the second last column have dashes all the way down (except for the first row).

So my code is going to have this form:

```
void middle(int row, int col, int startingNum, int size) {
    if (col == 1) {
        // first column
        printf("%d", (100+startingNum - 4*(size-1)%10 + row-1)%10);
    } else if (col == 2) {
        int someNumber = 100+startingNum - (4*(size-1))%10 + row-2;
        // calculate the only number in the second column
        middle2ndCol(row, someNumber, size); // function for second column
    } else if (col == size) {
        //last column
        printf("%d", (100+startingNum - (size-1)%10 - row%10 +1)%10);
    } else if (col == size - 1) {
        // second last column
        printf("-");
    } else {
        // Calculate new starting number for smaller box
        int newStartingNum = countNumbers(size-4) % 10-1;
        // Generate smaller box
        equations(row-2, col-2, newStartingNum, size-4);
    }
}
```

The function for the second column is below. Pretty much it will print `someNumber` in the third row i.e. the top (refer to the diagram) and print dashes along the rest of the column. The value of this number is calculated in the above function and fed as an argument below.

```
void middle2ndCol(int row, int someNumber, int size) {
    if (row == 3) {
        printf("%d", (someNumber)%10);
    } else {
        printf("-");
    }
}
```

7 Putting it all together

At the moment, we have wrote individual functions for different rows and columns. We can now start putting it all together now into this sub-main function called `equations()` which... as the same suggest... contains the equations for all of the functions we just wrote. It takes the row and column number as input and calculates the number that should go in that place. (it also takes startingnumber and size as inputs because we need those numbers to calculate the actual numbers)

```
void equations(int row, int col, int startingNum, int size) {
    if (row == 1) {
        // first row. see section 2
    } else if (row == 2) {
        // second row, see section 3
    } else if (row == size) {
        int lastRowStartNum = ... //see section 4
        rowLast(col, lastRowStartNum, size);
    } else if (row == size-1) {
        int lastRowStartNum = ... // see section 5
        row2ndLast(col, lastRowStartNum, size);
    } else {
        // calculate middle - see section 6
        middle(row, col, startingNum, size);
    }
}
```

8 Main function

The main function really just deals with the input.

```
int main(void) {
    int size;
    printf("Enter size: ");
    scanf("%d", &size);
    int countNum = countNumbers(size);
    int startingNum = countNum % 10 - 1;

    for (int row = 1; row <= size; row++) {
        for (int col = 1; col <= size; col++) {
            // print the character that should go in the position (row, col)
            equations(row, col, startingNum, size);
        }
        printf("\n"); // next row
    }

    return 0;
}
```