6주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181294 이름: 임승섭

**1.**

1 - 1. int CheckToMove

<예비보고서의 pseudo code>

for (int i = 0; i< 4; i ++)

for (int j = 0; j< 4; j++)

if currentblock[i][j] == 1

if (field[blockY + i][BlockX + j] == 1)

return 0;

else if (blockY + i >= HEIGHT)

return 0

else if (blockX + j < 0)

return 0

else if (blockX + j > WIDTH)

return 0

return 1

<실제 구현한 code>

int CheckToMove(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX){

    // user code

    for (int i = 0; i < BLOCK\_HEIGHT; i++) {

        for (int j = 0; j < BLOCK\_WIDTH; j++) {

            if (block[currentBlock][blockRotate][i][j] == 1) {

                if (blockY + i >= HEIGHT)

                    return 0;

                else if (blockX + j < 0)

                    return 0;

                else if (blockX + j >= WIDTH)

                    return 0;

                else if (f[blockY+i][blockX+j] == 1)

                    return 0;

            }

        }

    }

    return 1;

}

우선, block의 크기가 4 x 4 이기 때문에 i와 j의 범위를 단순히 4로 적었었는데, block 크기에 대한 변수가 따로 있는 것을 확인하고 i, j의 범위를 각 변수로 잡아주었다. 또한, 처음 pseudo code 를 작성할 때는 현재 block의 모양을 알려주는 2차원 배열 currentblock[i][j]가 있다고 생각하였지만, 실제 구현할 때에는 block[][][][] 배열을 이용하여 block의 모양을 확인하였다.

<시간복잡도>

block의 가로/세로를 각 n이라고 했을 때, 둘에 대해 for loop을 이중으로 돌기 때문에 시간복잡도는 O(n2) 이 된다.

<공간복잡도>

추가되는 메모리 크기 변화가 없기 때문에 공간복잡도는 O(1)이다.

1 - 2. void DrawChange

<예비보고서의 pseudo code>

for (int i = 0; i< 4; i++)

for (int j = 0; j< 4; j++)

field[blockY + i][blockX + j] = ‘.’

if (command == KEY\_UP)

DrawBlock(blockY, blockX, currentBlock, (blockRotate + 1)%4, 1)

else if (command == KEY\_DOWN)

DrawBlock(blockY+1, blockX, currentBlock, blockRotate, 1)

else if (command == KEY\_LEFT)

DrawBlock(blockY, blockX - 1, currentBlock, blockRotate, 1)

else if (command == KEY\_RIGHT)

DrawBlock(blockY, blockX + 1, currentBlock, blockRotate, 1)

<실제 구현한 code>

    int c\_Rotate = blockRotate;

    int c\_blockY = blockY;

    int c\_blockX = blockX;

    switch(command) {

        case KEY\_UP:

            c\_Rotate = (c\_Rotate+3)%4;

            break;

        case KEY\_DOWN:

            c\_blockY--;

            break;

        case KEY\_RIGHT:

            c\_blockX--;

            break;

        case KEY\_LEFT:

            c\_blockX++;

            break;

        default:

            break;

    }

    //2.

    for (int i = 0; i < BLOCK\_HEIGHT; i++) {

        for (int j = 0; j < BLOCK\_WIDTH; j++) {

            if ((block[currentBlock][c\_Rotate][i][j] == 1) && (i + c\_blockY>=0) ){

                move(c\_blockY + i + 1, c\_blockX + j + 1);

                printw(".");

            }

        }

    }

    //3.

    DrawBlock(blockY, blockX, currentBlock, blockRotate, ' ');

    move(HEIGHT, WIDTH+10);

실습 시간에 구현을 해보면서 처음 생각이 많이 잘못되었다는 것을 알았다. 우선, 함수의 parameter로 들어오는 변수들이 상태 변화 전이라고 생각하였다. 또한, field[][] 값을 ‘.’ 으로 변경하면 알아서 이전 block의 그림이 없어진다고 생각했다. 이를 수정하여, 먼저 c\_Rotate, c\_blockY, c\_blockX 라는 지역변수를 설정하여 block의 변화 이전의 상태를 파악하였다. move() 함수를 이용하여 이전의 block 위치로 이동했고, printw() 함수를 이용해서 직접 ‘.’을 찍어줌으로서 이전 block의 그림을 모두 지웠다. 마지막으로 실제 들어온 값을 이용해 현재 block을 DrawBlock() 함수로 그렸고, move() 함수를 이용해 커서를 밖으로 옮겼다.

<시간복잡도>

block의 가로/세로를 각 n이라고 했을 때, 둘에 대해 for loop을 이중으로 돌기 때문에 시간복잡도는 O(n2) 이 된다.

<공간복잡도>

추가되는 메모리 크기 변화가 없기 때문에 공간복잡도는 O(1)이다.

1 - 3. void BlockDown

<예비보고서의 pseudo code>

if (CheckToMove(field[HEIGHT][WIDTH], currentBlock, blockRotate, blockY+1, blockX) == 0)

if (blockY == -1)

gameover = 1

else

AddBlockToField(field[HEIGHT][WIDTH], currentBlock, blockRotate, blockY, blockX)

score += DeleteLine(field)

PrintScore(score)

nextBlock[0] = nextBlock[1]

nextBlock[1] = rand() %7

currentBlock = nextBlock[0]

blockRotate = 0

blockY = -1

blockX = (WIDTH/2) - 2

DrawBlock(blockY, blockX, currentBlock, 0, 1)

DrawNextBlock(nextBlock[1])

DrawField()

else

blockY++

DrawChange(field, KEY\_DOWN, currentBlock, blockRotate, blockY, blockX)

<실제 구현한 code>

    if (CheckToMove(field, nextBlock[0], blockRotate, blockY+1, blockX) == 0) {

        if (blockY == -1)

            gameOver = 1;

        else {

            AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX);

            score += DeleteLine(field);

            PrintScore(score);

            nextBlock[0] = nextBlock[1];

            nextBlock[1] = rand() %7;

            blockRotate = 0;

            blockY = -1;

            blockX = (WIDTH/2) - 2;

            DrawNextBlock(nextBlock);

        }

        DrawField();

    }

    else {

        blockY++;

        DrawChange(field, KEY\_DOWN, nextBlock[0], blockRotate, blockY, blockX);

    }

    timed\_out = 0;

크게 다른 점은 없다. 다만 BlockDown 함수는 매개변수로 현재 block을 받지 않기 때문에 currentBlock으로 썼던 부분을 모두 nextblock[0]으로 바꿔주었다. 맨 마지막에는 timed\_out을 0으로 초기화 해주었다.

<시간복잡도>

block의 가로/세로를 각 n이라고 했을 때, 우선 처음 CheckToMove 함수에서 O(n2)이 걸린다. 만약 true일 때, 일단 마지막에 DrawField에서 O(n2)이 걸린다. 여기까지 최소 O(n4)이고, blockY == -1 인 경우는 여기서 끝이다. 만약 그렇지 않으면, AddBlockToField, DrawNextBlock, DeleteLine이 모두 실행되고 이 세 함수의 시간복잡도는 모두 O(n2)이다. 서로 따로 실행되는 것이기 때문에 다 합쳐도 O(n2)이 유지된다. 결국, O(n6)이 걸리게 된다. true가 아닐 시에는 DrawChange의 O(n2) 때문에 총 O(n4)가 된다.

<공간복잡도>

추가되는 메모리 크기 변화가 없기 때문에 공간복잡도는 O(1)이다.

1 - 4. void AddBlockToField

<예비보고서의 pseudo code>

for (int i = 0; i< 4; i++)

for (int j = 0; j < 4; j++)

if (currentBlock[i][j] == 1)

field[blockY+i][blockX+j] = 1;

<실제 구현한 code>

    for (int i = 0; i < BLOCK\_HEIGHT; i++) {

        for (int j = 0; j < BLOCK\_WIDTH; j++) {

            if (block[currentBlock][blockRotate][i][j] == 1)

                f[blockY+i][blockX+j] = 1;

        }

    }

위와 마찬가지로 block의 크기를 표현할 때 4 x 4 대신 BLOCK\_HEIGHT와 BLOCK\_WIDTH 변수를 이용하여 표현하였고, 실제로 존재하지 않는 currentBLock[][] 배열 대신에 block[][][][] 배열을 이용하여 현재 block 의 상태를 확인하였다.

<시간복잡도>

block의 가로/세로를 각 n이라고 했을 때, 둘에 대해 for loop을 이중으로 돌기 때문에 시간복잡도는 O(n2) 이 된다.

<공간복잡도>

추가되는 메모리 크기 변화가 없기 때문에 공간복잡도는 O(1)이다.

1 - 5. int DeleteLine

<예비보고서의 pseudo code>

int DeadLine = 0

for (int i = 0; i< HEIGHT; i++)

for (int j = 0; j< WIDTH; j++)

if (field[i][j] == 0)

break;

if (j == WIDTH-1) &&(field[i][j] != 0)

for (int k = i; k>0; k--)

for (int t = 0; t < WIDTH; t++)

field[k][t] = field[k-1][t]

DeadLine++;

return (DeadLine)\*(DeadLine)\*100

<실제 구현한 code>

    int DeleteLine = 0;

    bool DeleteCheck = 0;

    for (int i = 0; i < HEIGHT; i++) {

        DeleteCheck = 1;

        for (int j = 0; j < WIDTH; j++) {

            if (f[i][j] == 0 ) {

                DeleteCheck = 0;

                break;

            }

        }

        if (DeleteCheck == 1) {

            DeleteLine++;

            for (int p = i; p > 0; p--) {

                for (int q = 0; q < WIDTH; q++) {

                    f[p][q] = f[p-1][q];

                }

            }

        }

    }

    return (DeleteLine)\*(DeleteLine) \* 100;

pseudo code를 작성할 때는 생각나는 대로 작성하다보니 따로 DeleteCheck 변수를 사용하지 않았다. 강의 자료와 ppt를 읽어보니 더 깔끔한 방법으로 구현할 수 있을 것 같아 전체적으로 내용을 많이 바꾸었다. 다만, pseudo code 상으로 구현하여도 크게 문제는 없어 보인다. 실제로 구현할 때에는 bool type의 DeleteCheck 변수를 이용하여 행 하나가 모두 다 1일 때를 확인하였다. if 조건문을 이용하여 모두 1이 될 때에는 DeleteLine을 +1 해주고, 다시 p와 q를 이용해서 모든 field의 내용을 한 줄씩 내려주었다. 마지막에는 score를 return 해주기 위해 DeleteLine \* DeleteLine \* 100을 return 해주었다.

<시간복잡도>

전체 field의 hieght와 width를 n이라고 했을 때, 일단 이중 loop을 돌기 때문에 O(n2)이다. 만약 DeleteCheck == 1 일 때, 또다시 이중 loop을 돌게 된다. 극단적으로 worst case일 때에는 여기서도 O(n2)이 걸리게 되고, 최종 시간복잡도는 O(n4)가 된다.

<공강복잡도>

추가되는 메모리 크기 변화가 없기 때문에 공간복잡도는 O(1)이다.

**2.**

2 - 1 DrawShadow

int sh\_y = y

while(1)

if chektomove(field, blockid, blockrotate, sh\_y+1, blockx))

sh\_y++

else

break

drawblock(sh\_y, blockx, blockid, blockrotate, ‘/’)

<시간복잡도>

전체 field의 size를 m x m, block의 size를 n x n이라고 하면, checktomove와 drawblock 함수의 시간복잡도는 O(n2) 이다. 만약 while문이 거의 실행되지 않는다면 O(n2)으로 끝나겠지만, 최악의 상황이라면 while문이 O(m)만큼 실행된다. 따라서 worst case에서는 시간복잡도가 O(n2 \* m)이 된다.

<공간복잡도>

추가되는 메모리 크기 변화가 없기 때문에 공간복잡도는 O(1)이다.

2 - 2 DrawBlockWithFeatures

DrawBlock(blockY, blockX, blockid, blockroatate, ‘ ‘);

DrawShadow(blockY, blockX, blockid, blockrotate);

<시간복잡도>

전체 field의 size를 m x m, block의 size를 n x n 으로 하면, 위에 보였듯이 DrawShadow의 시간복잡도는 O(n2 \* m)이고, DrawBlock의 시간복잡도는 O(n2)이다. 따라서 최종 시간복잡도는 O(n2 \* m)이다.

<공간복잡도>

추가되는 메모리 크기 변화가 없기 때문에 공간복잡도는 O(1)이다.

2 - 3 InitTetris

// change nextblock to size 3.

nextBlock[0] = rand() % 7

nextBlock[1] = rand() % 7

nextBlock[2] = rnad() % 7

// change DrawBlock to DrawBlockWIthFeatures

DrawBlockWithFeatures(blockY, blockX, nextBlock[0], blockRotate)

<시간복잡도>

원래 코드에서 변하지 않는다. O(m2)이 그대로 유지된다.

<공간복잡도>

마찬가지로 변하지 않는다. O(m2)이 그대로 유지된다.

2 - 4 drawnextblock

int i, j

for i = 0 to 4

move(4+i, width+13)

for j = 0 to 4

// draw nextblock[1] in the box

move (10 + i, width+13)

for j = 0 to 4

// draw nextblock[2] in the box

<시간복잡도>

j를 이용해 n만큼의 loop을 추가하긴 했지만 기존에 이미 있었기 때문에 시간복잡도는 O(n2)로 변하지 않는다.

<공간복잡도>

마찬가지로 변하지 않는다. O(1)이 유지된다.

2 - 5 blockDown

// change nextblock to size 3

nextBlock[0] = nextBlock[1]

nextBlock[1] = nextBlock[2]

nextBlock[2] = rand()%7

<시간복잡도&공간복잡도>

nextBlock 배열의 값만 변경하는 내용을 추가하였기 때문에 둘 다 변함이 없다. 위에 자세하게 쓴 내용이 있기 때문에 생략하였다.

2 - 6 addblocktofield

int touched = 0

for i = 0 to block\_height

for j = 0 to block\_width

if (f[blockY + i + 1][blockX + j] == 1)

touched++

f[blockY+i][blockX+j] = 1;

return touched \* 10

<시간복잡도>

원래 있던 loop 안에 조건문을 하나 추가하였기 때문에 이전과 변함이 없다.

<공간복잡도>

touched라는 변수가 추가되었지만 그 1개가 끝이기 때문에 마찬가지로 변함이 없다.

이 역시 위에 자세하게 설명하였기 때문에 생략하였다.

2 - 7 blockdown

// add the score from AddBlockToField

score += AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX)

<시간복잡도&공간복잡도>

원래 있는 변수 score, 원래 사용했던 함수 AddBlockToField이기 때문에 둘다 전혀 변함이 없다.