8주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

1.

모든 코드는 예비보고서에 작성한 pseudo 코드와 동일하게 구현되었다.

1-1. CheckToMove 함수

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) {

*// 게임판 범위 내에 블록이 존재한다면*

                if(i+blockY<0 || HEIGHT <= i+blockY || j+blockX<0 || WIDTH <= j+blockX || f[i+blockY][j+blockX] == 1) {

                    return 0;

                }

            }

        }

    }

    return 1;

}

시간복잡도: O(BLOCK\_HEIGHT\*BLOCK\_WIDTH), 상수만큼 반복되므로 O(1)이라고 할 수 있다.

공간복잡도: argument와 전역변수만을 사용했으므로 O(1)이다.

1-2. DrawChange 함수

void DrawChange(char f[HEIGHT][WIDTH],int command,int currentBlock,int blockRotate, int blockY, int blockX){

*// user code*

*//1. 이전 블록 정보를 찾는다. ProcessCommand의 switch문을 참조할 것*

    int ypos = 0, xpos = 0, rot = blockRotate;

    switch(command) {

        case KEY\_LEFT:

            xpos = 1;

            break;

        case KEY\_RIGHT:

            xpos = -1;

            break;

        case KEY\_UP:

            rot = (rot+3)%4;

            break;

        case KEY\_DOWN:

            ypos = -1;

            break;

        default:

            break;

    }

*//2. 이전 블록 정보를 지운다. DrawBlock함수 참조할 것.*

*// attroff(A\_REVERSE);*

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

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

            if(block[currentBlock][rot][i][j] == 1 && i+blockY>=0) {

*// 이전 블록의 위치 정보를 지운다.*

                move(i+blockY+ypos+1, j+blockX+xpos+1);

                printw(".");

*// 다음 방법으로도 똑같이 구현가능하지만,*

*// 전체 field를 다시 출력하는 것은 비효율적이므로*

*// 달라진 부분만을 갱신하는 위의 코드를 채택하였다.*

*// f[blockY+i+ypos][blockX+j+xpos] = 0;*

*// DrawField();*

            }

        }

    }

*//3. 새로운 블록 정보를 그린다.*

    DrawBlock(blockY, blockX, nextBlock[0], blockRotate,' ');

}

마지막에 호출되는 DrawBlock 함수

void DrawBlock(int y, int x, int blockID,int blockRotate,char tile){

    int i,j;

    for(i=0;i<4;i++)

        for(j=0;j<4;j++){

            if(block[blockID][blockRotate][i][j]==1 && i+y>=0){

                move(i+y+1,j+x+1);

                attron(A\_REVERSE);

                printw("%c",tile);

                attroff(A\_REVERSE);

            }

        }

    move(HEIGHT,WIDTH+10);

}

시간복잡도: O(BLOCK\_HEIGHT\*BLOCK\_WIDTH), 상수만큼 반복되므로 O(1)이라고 할 수 있다.

공간복잡도: argument와 전역변수를 사용하였고, 상수개의 변수를 사용하므로 O(1)이다.

1-3. BlockDown 함수

void BlockDown(int sig){

*// user code*

*//강의자료 p26-27의 플로우차트를 참고한다.*

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

        blockY++;

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

    }

    else {

        if(blockY == -1) {

            gameOver = 1;

        }

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

        score += DeleteLine(field);

        nextBlock[0] = nextBlock[1];

        nextBlock[1] = rand() % 7;

        blockRotate = 0;

        blockY=-1;

        blockX=WIDTH/2-2;

        DrawNextBlock(nextBlock);

        PrintScore(score);

        DrawField();

    }

*//BlockDown(sig);*

    timed\_out = 0;

}

시간복잡도:

앞에서 CheckToMove 함수의 시간복잡도는 O(1)이었다. DrawChange 함수 또한 O(1)이다. 그 외 등장하는 함수들이 모두 O(1)의 시간복잡도를 가지므로 O(1)이다.

1-4. AddBlockToField 함수

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

*// user code*

*//Block이 추가된 영역의 필드값을 바꾼다.*

    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;

            }

        }

    }

}

시간복잡도: O(BLOCK\_HEIGHT\*BLOCK\_WIDTH), 상수만큼 반복되므로 O(1)이라고 할 수 있다.

공간복잡도: argument와 전역변수가 사용되고, 상수개의 변수가 사용되므로 O(1)이라고 할 수 있다.

1-5. DeleteLine 함수

int DeleteLine(char f[HEIGHT][WIDTH]){

*// user code*

*//1. 필드를 탐색하여, 꽉 찬 구간이 있는지 탐색한다.*

    char isfull; *// 한 줄이 모두 채워졌는지 확인하는 변수(1: 모두 채워짐, 0: 모두 채워지지 않음)*

    int delete\_cnt = 0; *// 삭제한 줄의 개수*

    int delete\_ypos[HEIGHT+1]; *// 추가 배열을 이용해 시간복잡도 최소화*

    for(int i=HEIGHT-1; i>=0; i--) {

        isfull = 1;

        for(int j=0; j<WIDTH; j++) {

            if(f[i][j] == 0) {

                isfull = 0;

                break;

            }

        }

        if(isfull) { *// i번째 줄이 블록으로 꽉 차있다면*

            delete\_ypos[delete\_cnt] = i; *// i를 배열에 저장*

            delete\_cnt++; *// 삭제한 줄의 개수 증가*

        }

    }

    delete\_ypos[delete\_cnt] = 0;

*//2. 꽉 찬 구간이 있으면 해당 구간을 지운다. 즉, 해당 구간으로 필드값을 한칸씩 내린다.*

*// 꽉 찬 줄이 발견될 때마다 한 칸씩 아래로 이동시키는 것이 아니라,*

*// 지워야 할 줄을 delete\_ypos[]에 저장해 두었다가, 한 번에 아래로 이동시킨다.*

    for(int k=0; k<delete\_cnt; k++) {

        int target = delete\_ypos[k];

        for(int i=target-1; i>delete\_ypos[k+1]; i--) {

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

                f[i+k+1][j] = f[i][j];

            }

        }

        for(int j=0; j<WIDTH; j++) {

            f[0][j] = 0;

        }

    }

*// 삭제한 줄의 수를 점수로 환산하여 return*

    return delete\_cnt\*delete\_cnt\*100;

}

테트리스 칸을 탐색하다가 꽉 찬 줄이 발견되었을 때마다 전체를 한 줄씩 아래로 이동시키는 것이 아니라, 추가 배열을 선언하여 시간복잡도를 최소화하였다.

시간복잡도: O(HEIGHT\*WIDTH+HEIGHT\*WIDTH), 상수만큼 반복되므로 O(1)이라고 표현할 수 있다.

공간복잡도: argument와 전역변수를 사용하고 있으며 상수개의 변수를 사용하였으므로 O(1)이라고 말할 수 있다.

2.

각 기능을 구성하는 대표 함수의 시간복잡도와 공간복잡도에 대해 알아보았다.

2-1. 그림자 기능

void DrawShadow(int y, int x, int blockID,int blockRotate){

*// user code*

    int shadow\_ypos;

    for(shadow\_ypos=y; shadow\_ypos<HEIGHT; shadow\_ypos++) {

        if(CheckToMove(field, blockID, blockRotate, shadow\_ypos+1, x)==0) break;

    }

    DrawBlock(shadow\_ypos, x, blockID, blockRotate, '/');

}

시간복잡도: O(HEIGHT), 상수만큼 반복되므로 O(1)

공간복잡도: argument와 전역변수, 상수개의 변수가 사용되므로 O(1)

void DrawBlockWithFeatures(int y, int x, int blockID, int blockRotate) {

    DrawShadow(y, x, blockID, blockRotate);

    DrawBlock(y, x, blockID, blockRotate, ' ');

}

시간복잡도: O(1)

공간복잡도: argument만 사용되므로 O(1)

2-2. 2개의 블록 미리 보여주기

void DrawNextBlock(int \*nextBlock){

    int i, j;

    for( i = 0; i < 4; i++ ){

        move(4+i,WIDTH+13);

        for( j = 0; j < 4; j++ ){

            if( block[nextBlock[1]][0][i][j] == 1 ){

                attron(A\_REVERSE);

                printw(" ");

                attroff(A\_REVERSE);

            }

            else printw(" ");

        }

        move(10+i, WIDTH+13);

        for(j=0; j<4; j++) {

            if(block[nextBlock[2]][0][i][j] == 1) {

                attron(A\_REVERSE);

                printw(" ");

                attroff(A\_REVERSE);

            }

            else printw(" ");

        }

    }

}

시간복잡도: O(BLOCK\_HEIGHT\*(BLOCK\_WIDTH\*2))이므로 O(1)

공간복잡도: argument와 전역변수, 상수개의 변수를 사용하므로 O(1)

void BlockDown(int sig){

*~*

        nextBlock[0] = nextBlock[1];

        nextBlock[1] = nextBlock[2];

        nextBlock[2] = rand() % 7;

~

}

시간복잡도: 1-3와 동일한 이유로 O(1)

시간복잡도: 1-3과 동일한이유로 O(1)

2-3. 닿은 면적만큼 score 증가하기

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

*// user code*

*//Block이 추가된 영역의 필드값을 바꾼다.*

    int bottom\_cnt = 0;

    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-1) bottom\_cnt++;

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

                    bottom\_cnt++;

                }

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

            }

        }

    }

    return bottom\_cnt\*10;

}

시간복잡도: O(BLOCK\_HEIGHT\*BLOCK\_WIDTH), O(1)

공간복잡도: argument와 전역변수, 상수개의 변수를 사용하므로 O(1)