8 주차 결과보고서

전공: 컴퓨터공학과 학년 2 학년 학번 20231632 이름: Jumagul Alua

1. 프로그램의 함수들이 pseoudo code랑 어떻게 달라졌는지, 그리고 시간 및 공간 복잡도

1) CheckToMove():

가장 큰 차이는 루프에서 사용되는 변수들의 이름과 구조체 블록을 다루는 방식이다. 여기서 공간 복잡도는 상수값에 의해 결정된다. 프로그램에서 사용된 추가적인 변수들은 상수이므로 공간 복잡도는 O(1)이다.

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

for i = 0 to BLOCK_HEIGHT:

for j = 0 to BLOCK_WIDTH:

if block[currentBlock][blockRotate][i][j]가 1이고 (i+blockY, j+blockX) 필드를 벗어남or이미 블록이 있다면

return 0

return 1

```
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) {
                int blY=i+blockY, blX=j+blockX;
                //field 벗어나지 않는지 확인
                if(blY>=HEIGHT || blX>=WIDTH || blX<0){
                    return 0;
                }
                //이미 차있는 곳에 놓이지 않았는지 확인
                if(f[blY][blX]==1) {
                    return 0;
                }
            }
        }
    }
    return 1;
}
```

2) DrawChange()

두 코드의 차이점 중 하나는 변수 이름과 그들이 사용된 방식이다. 작성한 코드에는 블록의 현재 위치를 지운 다음, 새로운 위치에 블록을 그리는 데 사용되는 루프가 있다. 이 루프는 pseudo code 의 이중 루프가 아니라 단일 루프로 구성되어 있다. 그리고 'for i,j=0 to 3'이라고 pseudo code 에 정의했지만, 블록 너비와 넓이가 3 이 아닌 4 라서, BLOCK_HEIGTH,

BLOCK_WEIDTH 로 바꿨다. 루프는 블록의 크기에 따라 결정되므로 시간 복잡도는 O(BLOCK_HEIGHT*BLOCK_WIDTH)이다. 공간 복잡도는 추가적인 변수와 상수만 사용되므로 여전히 O(1)이다.

```
DrawChange(char f[HEIGHT][WIDTH],int command,int currentBlock,int blockRotate, int blockY, int blockX)
    preRot = blockRotate
    preblX = blockX
    preblY = blockY
    flag = 1
        case KEY_UP preRot← (preRot + 3) % 4
       case KEY_DOWN preblY ← preblY - 1
       case KEY_LEFT preblX ← preblX + 1
       case KEY_RIGHT preblX ← preblX - 1
       default break
     for i=0 to 3
       for j=0 to 3
          if block[currentBlock][preRot][i][j] = 1
             기존의 그림자를 지운다
    DrawBlockWithFeatures(blockY, blockX, currentBlock, blockRotate)
```

```c
void DrawChange(char f[HEIGHT][WIDTH],int command,int currentBlock,int blockRotate, int blockY, int blockX){
    int pre_blX=blockX, pre_blY=blockY, pre_Rot=blockRotate; //바뀌기 전 현재 블록의 정보를 얻어서 switch문 사용
    switch(command){
        case KEY_UP:
            pre_Rot=(pre_Rot+3)%4;
            break;
        case KEY_DOWN:
            pre_blY -= 1;
            break;
        case KEY_RIGHT:
            pre_blX -= 1;
            break;
        case KEY_LEFT:
            pre_blX += 1;
            break;
        default:
            break;
    }//아전에 그려진 현재 블록을 화면에서 지우기
    for(int i=0; i<BLOCK_HEIGHT; i++) {
        for(int j=0; j<BLOCK_WIDTH; j++) {
            if(block[currentBlock][pre_Rot][i][j]==1 && pre_blY>=0) {
                move(i+pre_blY+1, j+pre_blX+1);
                printw(".");
            }
        }
    }//현재 블록을 화면에 그려주기
    DrawBlock(blockY, blockX, nextBlock[0], blockRotate,' '); //현재 블록을 화면에 그려주기
    move(HEIGHT, WIDTH+10); //커서를 필드 밖으로 이동하기
```

3) *BlockDown()*

두 코드의 반복문은 블록의 크기에 비례하므로, 시간 복잡도는 O(BLOCK_HEIGHT*BLOCK_WIDTH)이다. 공간복잡도도 O(1)이다.

AddBlockToField 함수에score 을 업데이트하지는 않았고, "nextBlock[0] ← nextBlock[1] nextBlock[1] ← nextBlock[2]" 대신 "for(int i=0; i<2; i++) nextBlock[i]=nextBlock[i+1]로 바꿨고, timed_out 을 0 으로 정의해줬다.

```
BlockDown(sig)
  if CheckToMove(field, nextBlock[0], blockRotate, blockY + 1, blockX)
    for i=0 to BLOCK_HEIGHT
      for i=0 to BLOCK_WIDTH
        If block[currentBlock][blockRotate][i][j] = 1
          f[blockY + i][blockX + j] = 1
        if blockY ≤ -1
        gameOver = 1
        return
      score += AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX)
      score += DeleteLine(field)
      nextBlock[0] ← nextBlock[1]
```

nextBlock[1] ← nextBlock[2]

nextBlock[2] = rand() % 7

blockRotate = 0

blockY = -1

blockX = WIDTH / 2 - 2

DrawField()

DrawNextBlock(nextBlock)

PrintScore(score)

```
void BlockDown(int sig){
    if(CheckToMove(field, nextBlock[0], blockRotate, blockY+1, blockX)) {//checktomove 함수를 호출하여 내려갈 수 있는지 확인
        for(int i=0; i<BLOCK_HEIGHT; i++) {
            for(int j=0; j<BLOCK_WIDTH; j++) {
                if(block[nextBlock[0]][blockRotate][i][j]==1 && blockY+i>=0) {
                    move(blockY+i+1, blockX+j+1);
                    printw(".");
                    }
                }
        }
        blockY++;//내려갈 수 있으면 y 좌표를 증가시켜주기
        DrawBlock(blockY, blockX, nextBlock[0], blockRotate, ' ');// 한 줄 내린 위치에 다시 그려주기

    } else {
        ifsame=0;
        AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX);
        if (blockY== -1){ //블록의 y좌표가 -1일 경우
            gameOver=1;
        }
        else {
            AddBlockToField(field,nextBlock[0], blockRotate, blockY, blockX); //블록을 field에 합치기
            score+=DeleteLine(field); //line을 지우고 score를 업데이트하기

            for(int i=0; i<2; i++)
                nextBlock[i]=nextBlock[i+1];
            nextBlock[2]=rand()%7;
            DrawNextBlock(nextBlock);
            //초기화
```

```
            nextBlock[2]=rand()%7;
            DrawNextBlock(nextBlock);
            //초기화
            blockY=-1;
            blockX=WIDTH/2-2;
            blockRotate=0;

            PrintScore(score);
        }
        DrawField(); //화면에 출력
    }
    timed_out=0; //play에서 timed_out이 0일 때 호출됨
}
```

4) *AddBlockToField*

Pseudo code 에는 단순한 조건문만 포함되어 있지만, 작성한 코드에는 추가적인 조건과 변수를 활용하여 필드에 블록을 추가하는 방법이

더욱 세분화되어졌다. 주어진 배열 f 에만 작업을 수행하므로, 공간 복잡도는 O(1)이고 시간 복잡도도 O(1)이다.

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

```
for i=0 to 3
    for j=0 to 3
        if block[currentBlock][blockRotate][i][j] = 1
            f[blockY + i][blockX + j] = 1
```

```
void AddBlockToField(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX){
    // user code
    int i, j, touched=0;
    for(i=0; i<BLOCK_HEIGHT; i++){
        for(j=0; j<BLOCK_WIDTH; j++) {
            if(block[currentBlock][blockRotate][i][j]==1) {
                if(blockY+i==HEIGHT-1){
                    touched++;
                }
                else if(f[blockY+i+1][blockX+j]==1) touched++;
                f[blockY+i][blockX+j]=1;
            }
        }
    }
}
```

5) DeleteLine

거의 달라진 점이 없고, 시간 복잡도는 O(HEIGHT * WIDTH^2)이고, 공간 복잡도는 O(1)이다.

DeleteLine(char f[HEIGHT][WIDTH])

```
line_cnt = 0
    for i from 0 to HEIGHT-1
        for j from 0 to WIDTH-1
            if f[i][j] = 0
                isFullLine ← false
                break
        if isFullLine
            line_cnt ++
        for k=j down to 1
            for j=0 to WIDTH-1
                f[k][j] = f[k-1][j]
```

return line_cnt^2 * 100

```
int DeleteLine(char f[HEIGHT][WIDTH]){
    int line_cnt=0;
    //int isFullLine=1;
    for(int i=0; i<HEIGHT; i++) {
        int isFullLine=1;
        for(int j=0; j<WIDTH; j++) {
            if(f[i][j]==0 ){
                isFullLine=0;
                break;
            }
        }//한 줄이 채웠는지 check
        if(isFullLine) {
            line_cnt++;
            //블록의 정보를 한 줄씩 내려주기
            for(int k=i; k>0; k--) {
                for(int l=0; l<WIDTH; l++) {
                    f[k][l]=f[k-1][l];
                }
            }
            for(int l=0; l<WIDTH; l++) {
                f[0][l]=0;
            }
        }
    }
    //점수 ((지운 line 개수)^2*100) return
    return line_cnt * line_cnt * 100;
}
```

2. 1 주차 숙제 문제를 해결하기 위한 pseudo code

(a) DrawShadow(int y, int x, int blockID, int blockRotate):

```
if ifsame != 1:
    for shadowY from y to HEIGHT:
        if CheckToMove(field, blockID, blockRotate, shadowY + 1, x) == 0:
            break
    DrawBlock(shadowY, x, blockID, blockRotate, '/')
ifsame = 0
```

시간 복잡도는 O(HEIGHT - y)이고 공간 복잡도는 O(1)이다.

(b) DrawBlockWithFeatures(int y, int x, int blockID, int blockRotate):

```
DrawShadow(y, x, blockID, blockRotate)
DrawBlock(y, x, blockID, blockRotate, ' ')
```

시간 복잡도는 O(HEIGHT - y)이고 공간 복잡도는 O(1)이다.