8주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20192135 이름: 윤영인

**1.**

1) CheckToMove

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Pseudo code로 작성한 코드와 크게 다른 점이 없으며 정상적으로 실행되었다. 해당 코드의 시간 복잡도는 O(블록의 크기 \* 블록의 크기) = O(4 \* 4) = O(16)이다. 즉, for문이 상수 번 반복되는 상수 시간 복잡도이므로 O(1)이다. 공간 복잡도는 사용되는 변수가 고정된 크기의 배열과 상수를 사용하므로 O(1)이다.

2) DrawChange

텍스트, 전자제품, 디스플레이, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Pseudo code와 달라진 점은 이전 좌표를 구하기 위해 pX, pY, pR 변수를 선언하여 현재 좌표의 정보를 저장하였다는 것이다. 또한 command 값에 따라 선언한 변수를 후위 연산자를 사용하여 값을 변경하였다. for문을 돌면서 해당 좌표로 이동해서 블록을 지울 때 pY + i >= 0을 확인하여 필드를 벗어나는지 확인하였다. 해당 코드의 시간 복잡도는 O(블록의 크기 \* 블록의 크기) = O(4 \* 4) = O(16)이다. 즉, for문이 상수 번 반복되는 상수 시간 복잡도이므로 O(1)이다. 공간 복잡도는 사용되는 변수가 고정된 크기의 배열과 상수를 사용하므로 O(1)이다.

3) BlockDown

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

Pseudo code와 달라진 점은 없으며 시간 복잡도는 O(1)이며, 공간 복잡도는 상수 변수만 사용하므로 O(1)이다.

4) AddBlockToField

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

Pseudo code와 다른 점은 블록이 이미 쌓여 있을 때 좌표가 필드 영역을 벗어나는지 확인하는 if문을 추가했다는 것이다. 시간 복잡도는 O(블록의 크기 \* 블록의 크기) = O(4 \* 4) = O(16)이다. 즉, for문이 상수 번 반복되는 상수 시간 복잡도이므로 O(1)이다. 공간 복잡도는 사용되는 변수가 고정된 크기의 배열과 상수를 사용하므로 O(1)이다.

5) DeleteLine

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

Pseudo code와 다르게, j == WIDTH이면 한 칸이라도 비어있지 않고 for(j = 0; j < WIDTH; j++)를 다 돌았다는 의미, 즉 한 줄이 꽉 차 있는 것을 의미하므로 꽉 찬 줄을 세는 변수를 하나 추가한다. 그리고 한 칸씩 아래로 내려서 꽉 찬 줄을 삭제한다. 시간 복잡도는 O(블록의 크기 \* 블록의 크기) = O(4 \* 4) = O(16)이다. 즉, for문이 상수 번 반복되는 상수 시간 복잡도이므로 O(1)이다. 공간 복잡도는 사용되는 변수가 고정된 크기의 배열과 상수를 사용하므로 O(1)이다.

**2.**

1) DrawBlockWithFeatures

DrawBlock()

DrawShadow()

-> 시간 복잡도: O(1), 공간 복잡도: O(1)

2) DrawShadow

int shadowy = 현재 y좌표 + 1

while shadowy < HEIGHT) {

if ((shadowy, x)좌표로 이동할 수 있는지 확인) {

shadowY++

}

}

shadow--

DrawBlock()

-> 시간 복잡도: O(1), 공간 복잡도: O(1)

3) BlockDown

score += AddBlockToField

nextBlock[0] = nextBlock[1]

nextBlock[1] = nextBlock[2]

nextBlock[2] = rand() % 7

를 추가한다.

-> 시간 복잡도: O(1), 공간 복잡도: O(1)

4) DrawNextBlock

2번째 next block을 그리기 위해,

for(i = 0 ~ 4) {

(10 + i, WIDTH + 13)좌표로 이동

for(j = 0 ~ 4) {

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

채워진 공백을 그림

}

else 공백을 그림

} 을 추가한다.

-> 시간 복잡도: O(1), 공간 복잡도: O(1)

5) InitTetris

nextBlock[2] = rand() % 7

DrawBlockWithFeatures()를 추가한다.

-> 시간 복잡도: O(1), 공간 복잡도: O(1)

6) AddBlockToField

int touched = 0

for (i = 0 ~ 4) {

for (j = 0 ~ 4) {

if(현재 좌표에서 block이 존재하면) {

if (현재 좌표가 필드를 벗어나지 않으면) {

f[y][x] = 1

if(f[y + 1][x] || (y == 바닥)) touched 1 증가

}

}

}

}

-> 시간 복잡도: O(1), 공간 복잡도: O(1)