8주차 결과보고서

전공: 경제학과 학년: 4학년 학번: 20180501 이름: 김연수

9-3. 결과 보고서

아래의 사항을 작성하여 다음 실험 시간에 제출하시오.

1. 실습시간에 작성한 프로그램의 함수들이 예비보고서에서 작성한 각 구현 함수들의 pseudo code와 어떻게 달라졌는지 설명하고, 각 함수에 대한 시간 및 공간 복잡 도를 보이시오(각 함수의 시간 및 공간복잡도를 구할 때, 어떤 변수에 의존하는지 를 판단해야 한다).

2. 테트리스 프로젝트 1주차 숙제 문제를 해결하기 위한 pseudo code를 기술하고, 작 성한 pseudo code의 시간 및 공간 복잡도를 보이시오.

1. 1번

CheckToMove:

1. int CheckToMove(char f[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX){
2. int i, j;
3. for (i = 0; i < 4; i++) {
4. for (j = 0; j < 4; j++) {
5. // Check if the current block cell is occupied
6. if (block[currentBlock][blockRotate][i][j]) {
7. // Check if the new position is within the field boundaries
8. if (blockY + i >= HEIGHT || blockY + i < 0 || blockX + j >= WIDTH || blockX + j < 0) {
9. return 0; // Can't move or rotate as expected
10. }
11. // Check if the new position is not occupied by another block
12. if (f[blockY + i][blockX + j]) {
13. return 0; // Can't move or rotate as expected
14. }
15. }
16. }
17. }
18. return 1; // The block can be moved or rotated as expected
19. }

pseudo code의 내용에서 변한 것이 거의 없다. pseudocode를 구현한 C code의 내용은 아래와 같다.

* 4\*4의 범위 안에서 i, j 의 위치에서 블럭이 존재하는 지 확인

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

* 새로운 포지션이 field의 밖을 벗어나지 않는지 확인하는 코드 :

if (blockY + I >= HEIGHT || blocky + I < 0 || blockX +j >= WIDTH || blockX + j < 0)

* 새로운 포지션이 다른 블럭에 의해 이미 점유되어 있는 경우

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

시간복잡도: 4\*4만큼 loop를 돌기 때문에 상수의 시간 복잡도를 가진다. -> O(1)

공간복잡도: 이 함수에서 할당하는 굥간이 상수 개수이므로 O(1)이다.

DrawChange:

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

int prevY = blockY;

int prevX = blockX;

int prevRotate = blockRotate;

int i, j;

// Determine the previous block position and rotation based on the command

switch (command) {

case KEY\_UP:

prevRotate = (blockRotate + 3) % 4;

break;

case KEY\_DOWN:

prevY--;

break;

case KEY\_RIGHT:

prevX--;

break;

case KEY\_LEFT:

prevX++;

break;

default:

break;

}

/\* Erase the previous block

\* 이중 포문을 사용할 것임(drawblock함수 참고)

\* currentBlock이 현재 블락의 id를 나타냄.

\*/

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

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

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

move(i+prevY+1,j+prevX+1);

printw("%c",'.');

}

}

// Draw the new block

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

}

pseudo code의 내용에서 변한 것이 거의 없다. pseudocode를 구현한 C code의 내용은 아래와 같다.

* Determine the previous block position and rotation based on the command

switch (command) {

case KEY\_UP:

prevRotate = (blockRotate + 3) % 4;

break;

case KEY\_DOWN:

prevY--;

break;

case KEY\_RIGHT:

prevX--;

break;

case KEY\_LEFT:

prevX++;

break;

default:

break;

}

* draw new block with DrawBlcok function

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

시간복잡도: command를 확인하는 부분, 이전 block으 지우는 부분, drawblock함수 역시 모두 상수의 시간복잡도를 가진다. -> O(1)

공간복잡도: 할당하는 공간이 상수이므로 -> O(1)

BlockDown:

void BlockDown(int sig){

//블럭을 더 이상 내릴 수 없는 경우

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

//blockY의 좌표가 -1인 경우 블럭이 끝까지 쌓인 것을 의미 -> 게임 종료 flag = True

if(blockY == -1) gameOver = 1;

//블럭을 field에 합친다.

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

//완전히 채워진 line을 지운다.

score += DeleteLine(field);

//nextblock을 currentblock으로 바꿔준다.

nextBlock[0] = nextBlock[1];

nextBlock[1] = rand() % 7;

DrawNextBlock(nextBlock);

PrintScore(score);

//initialize Location of current block

blockY = -1;

blockX=WIDTH/2-2;

blockRotate = 0;

DrawField();

} else {

blockY++;

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

}

timed\_out = 0;

}

pseudo code의 내용에서 변한 것이 거의 없다. pseudocode를 구현한 C code의 내용은 아래와 같다.

* add block to field with the function of addblocktofield.

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

* Erase Line by DeleteLine function. and add the return value in score.

score += DeleteLine(field);

* replace currentblock to nextblock and nextblock to rand number 0~6

nextBlock[0] = nextBlock[1];

nextBlock[1] = rand() % 7;

* initialize location of current block

blockY = -1;

blockX=WIDTH/2-2;

blockRotate = 0;

시간복잡도: 가장 timecomplexity가 심한 것이 deleteLIne() 또는 Drawfiled()일 것이다. 둘 다 O(HEIGHT \* WIDTH)의 시간복잡도를 가지므로 blockDown의 최악의 경우 시간복잡도 또한 O(HEIGHT \* WIDTH)이다.

공간복잡도: O(1). 상수 개수만큼 공간을 할당한다.

AddBlockToField:

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

//blockY, blockX는 현재 블럭의 위치.

//현재 블럭의 위치부터 시작해서

//block에서 i,j 2중 for loop를 돈다.

//그러면서 field에 blockY, blockX 좌표에 +i, +j를 한 위치에다가 블락의 값이 1인 경우 1을 넣어준다.

int i, j;

// Iterate through the 4x4 block matrix

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

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

// If the block has a non-zero value at the current position, add it to the field

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

// Check if the block is within the field boundaries before adding it

if (blockY + i >= 0 && blockY + i < HEIGHT && blockX + j >= 0 && blockX + j < WIDTH) {

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

}

}

}

}

}

pseudo code의 내용에서 변한 것이 거의 없다. pseudocode를 구현한 C code의 내용은 아래와 같다.

* Check if the block is within the field boundaries before adding it

f (blockY + i >= 0 && blockY + i < HEIGHT && blockX + j >= 0 && blockX + j < WIDTH)

시간복잡도: 4\*4 -> O(1)

공간복잡도: O(1)

DeleteLine:

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

int i, j, k;

int linesDeleted = 0;

int isLineFull;

// Iterate through each row in the field

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

isLineFull = 1;

// Check if the current row is full

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

if (f[i][j] == 0) {

isLineFull = 0;

break;

}

}

// If the row is full, delete it and move rows above it down

if (isLineFull) {

linesDeleted++;

// Move all rows above the deleted row down by 1

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

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

f[k][j] = f[k - 1][j];

}

}

// Clear the top row

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

f[0][j] = 0;

}

}

}

return (linesDeleted \* linesDeleted) \* 100;

}

pseudo code의 내용에서 변한 것이 거의 없다. pseudocode를 구현한 C code의 내용은 아래와 같다.

* return linesDeleted\*100

return linesDeleted \* linesDeleted \* 100;

시간 복잡도 : O(HEIGHT \* WIDTH)이다. 세로길이 만큼 위에서 부터 차례대로 탐색하면서 블락으로 다 채워진 라인이 존재하는 지 살핀다. 2차원 filed배열을 모두 탐색해야 하므로 시간 복잡도는 O(HEIGHT \* WIDTH)이다.

공간복잡도: O(1)이다. 상수만큼 공간을 할당했다.

2. 과제 함수의 pseudo code를 기술, 시간복잡도와 공간 복잡도

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

find the position where the block can no longer move down and let shadowy is y of that position.

Draw the block at the shadow position using '/' as the tile character by call DrawBlock function

}}

시간 복잡도: find shadowy -> O(HEIGHT). 따라서 DrawShadow의 시간 복잡도 또한 O(HEIGHT)이다.

공간 복잡도: O(1)

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

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

DrawShadow(y, x, blockID, blockRotate);

}

시간 복잡도 : drawblock.->상수시간 복잡도, drawShadow -> O(HEIGHT). 따라서 DrawBlockWithFeatures의 시간복잡도는 O(HEIGHT)

공간 복잡도: 두 함수의 공간 복잡도 모두O(1) 이므로 drawblockFeatures 또한 O(1)

(3) 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);

}

시간 복잡도: O(1)

공간 복잡도: O(1)