11주차 결과보고서

전공 : 컴퓨터공학 학년 : 2학년 학번 : 20151616 이름 : 최승환

1. 실습 시간에 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술하시오. 완성한 알고리즘의 시간 및 공간 복잡도를 보이시오.

a. recommendedPlay 함수에서는 act.sa\_handler 에 함수 recommender를 사용한다. 이렇게 되면 몇 초가 지난 후에 alarm()을 통해 특정 action을 취하게 되는데 이 action을 recommender 함수로 대신하게 되는 것이다. 쉽게 말해, 1초에 한번씩 recommender 함수가 호출되는 것이다. 그 이외의 점은 play 함수와 거의 유사하다.

* recommendedPlay 함수의 코드

void recommendedPlay(){

int command;

clear();

act.sa\_handler = recommender;

sigaction(SIGALRM,&act,&oact);

InitTetris();

do

{

if(timed\_out == 0)

{

alarm(1);

timed\_out = 1;

}

command = GetCommand();

if(ProcessCommand(command) == QUIT)

{

alarm(0);

DrawBox(HEIGHT/2-1,WIDTH/2-5,1,10);

move(HEIGHT/2,WIDTH/2-4);

printw("Good-bye!!");

refresh();

getch();

return;

}

}while(!gameOver);

alarm(0);

getch();

DrawBox(HEIGHT/2-1,WIDTH/2-5,1,10);

move(HEIGHT/2,WIDTH/2-4);

printw("GameOver!!");

refresh();

getch();

}

b . recommender 함수는 가중치를 이용해 해당 필드의 적합도를 판단하는 함수이다. 이 함수에서는 x좌표를 하나씩 늘려가며 현재 필드에 해당하는 x좌표에서 블럭이 내려갔을 경우의 정보를 newfield에 저장되고 이 newfield를 분석해 적합도를 판단한 후 가장 적합한 newfield의 상태를 저장해 field에 저장한다.

따라서 이 함수는 각 회전(4 경우)에 대해서 필드를 한 번씩 복사해오므로 4n3 를 조금 초과한다고 할 수 있다. 따라서 이 함수의 시간 복잡도는 O(n3)이라 할 수 있다.

* recommender 함수의 코드

void recommender(int sig/\*RecNode \*root\*/){

int maxy=0,maxx=0,maxc=0,maxb=0; // 미리 보이는 블럭의 추천 배치까지 고려했을 때 얻을 수 있는 최대 점수

int rcnt,ccnt=0,bcnt;

int flag=0;

int height=0,bumpiness=0,holes=0,completelines=0;

int r2=0,c2=0;

char newfield[HEIGHT][WIDTH];

double maxscore=0;

double a=-0.510066;

double b = 1.760666;

double c = -0.55663;

double d = -0.184483;

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

else

{

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

{

for(ccnt=0;ccnt<WIDTH;++ccnt)

{

height=0;

holes=0;

completelines=0;

for(r2=0;r2<HEIGHT;++r2)

{

for(c2=0;c2<WIDTH;++c2)

{

newfield[r2][c2] = field[r2][c2];

}

}

//cpfield(newfield);

rcnt=0;

while(CheckToMove(newfield,nextBlock[0],bcnt,++rcnt,ccnt));

--rcnt;

AddBlockToField(newfield,nextBlock[0],bcnt,rcnt,ccnt);

height = checkheight(newfield,&bumpiness);

holes = checkholes(newfield);

completelines = checkcompletelines(newfield);

if(flag == 0)

{

flag = 1;

maxscore = a\*height+b\*completelines+c\*holes+d\*bumpiness;

}

if(maxscore<a\*height+b\*completelines+c\*holes+d\*bumpiness)

{

maxscore = a\*height+b\*completelines+c\*holes+d\*bumpiness;

maxx = ccnt;

maxy = rcnt;

maxb = bcnt;

}

}

}

for(r2=0;r2<HEIGHT;++r2)

{

for(c2=0;c2<WIDTH;++c2)

{

newfield[r2][c2] = field[r2][c2];

}

}

}

if(maxx == 0)

{

rcnt=0;

while(CheckToMove(field,nextBlock[0],maxb,++rcnt,maxx));

--rcnt;

score+=AddBlockToField(field,nextBlock[0],maxb,rcnt,maxx);

}

else

{

score+=AddBlockToField(field,nextBlock[0],maxb,maxy,maxx);

}

score+=DeleteLine(field);

PrintScore(score);

DrawField();

nextBlock[0] = nextBlock[1];

nextBlock[1] = nextBlock[2];

nextBlock[2] = rand()%7;

DrawNextBlock(nextBlock);

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

timed\_out = 0;

return;

}

* 가중치 알고리즘에 필요한 height, holes, bumpiness, completelines를 구하는 각각의 함수 checkheight, checkholes checkcompletelines의 코드

int checkheight(char newfield[HEIGHT][WIDTH],int\* bumpiness)

{

int ccnt,rcnt;

int height=0,befheight=0,maxh=0;

\*bumpiness = 0;

for(ccnt=0;ccnt<WIDTH;++ccnt)

{

befheight = height;

height=0;

for(rcnt=0;rcnt<HEIGHT;++rcnt)

{

if(newfield[rcnt][ccnt] == 1)

{

break;

}

}

height = HEIGHT-rcnt;

if(height-befheight > 0)

{

\*bumpiness += (height-befheight);

}

else

{

\*bumpiness -= (height-befheight);

}

if(maxh<height)

{

maxh = height;

}

}

return maxh;

}

int checkholes(char newfield[HEIGHT][WIDTH])

{

int ccnt,rcnt,flag=0;

int holes=0;

for(ccnt=0;ccnt<WIDTH;++ccnt)

{

flag=0;

for(rcnt=0;rcnt<HEIGHT;++rcnt)

{

if(newfield[rcnt][ccnt])

{

flag=1;

}

else if(flag == 1)

{

//flag=0;

holes++;

}

}

}

return holes;

}

int checkcompletelines(char newfield[HEIGHT][WIDTH])

{

int delete\_num=0;

int ycnt,xcnt,clear\_flag;

for(ycnt=0;ycnt<HEIGHT;++ycnt)

{

clear\_flag=1;

for(xcnt=0;xcnt<WIDTH;++xcnt)

{

if(newfield[ycnt][xcnt] == 0)

{

clear\_flag=0;

break;

}

}

if(clear\_flag == 1)

{

++delete\_num;

}

}

return delete\_num;

}