|  |
| --- |
|  |
| **컴퓨터공학실험1 (1반) 11주차 결과보고서 (테트리스(3))** |
|  |
|  |
|  |
| 컴퓨터공학과 20161620 이수연 |

**컴퓨터공학실험1 // 테트리스(3) 결과보고서**

전공: 컴퓨터공학과 학년: 2학년 학번: 20161620 이름: 이수연

**1. 실습 시간에 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술하시오. 완성한 알고리즘(추가 구현하게 되는 효율성을 고려한 tree도 포함)의 시간 및 공간복잡도를 보이시오.**

가능한 모든 경우의 수를 따져봐야 하기 때문에 recommend 기능은 tree 자료구조를 이용할 수 있다.

<tree를 구성하는 node>

typedef struct \_RecNode{

int lv,score;

char f[HEIGHT][WIDTH];

struct \_RecNode \*c[CHILDREN\_MAX];

struct \_RecNode \*parent;

int curBlockID;

int recBlockX,recBlockY,recBlockRotate;

int countall;

}RecNode;

Tree를 구성할 때는 이중 반복문을 사용한다. 따라서 시간복잡도는 r[nextBlock[root->depth+1]] x WIDTH 이다. 또한 하나의 반복문 안에서 계속해서 child를 생성하는 구조이므로, node의 수인 공간복잡도 또한 시간복잡도와 같을것이다.

**2. 모든 경우를 고려하는 tree 구조와 비교해서 어떤 점이 더 향상되고, 어떤 점이 그렇지 않은지 아울러 기술하시오.**

가장 높은 점수를 계산하기 위해 프로그램의 수행할수록 필요한 시간 복잡도와 공간복잡도가 커져 프로그램의 진행이 느려진다는 큰 단점이 있다. 가령 이 프로그램만 보았을 때도 모든 경로를 탐색하기 위한 시간 복잡도는 O(34^n)이 되기 때문에, 경우의 수가 증가하면 시간복잡도가 기하급수적으로 증가하여 비효율적이다. 실제 play시에도 게임의 진행이 느려지게 된다. 공간복잡도의 측면에서도 이와 마찬가지로 경우의 수가 증가할수록 많은 공간이 사용되기 때문에 비효율적이며 메모리 overflow가 발생될 수도 있다. 이를 해결하기 위해서는 n을 줄이는 방법, Pruning tree를 이용하는 방법(시간의 비효율성 해결), Data Simplication(공간의 비효율성 해결)을 이용하는 방법 등을 사용할 수 있다.

**3. 테트리스 프로젝트 3주 과정을 통해 습득한 내용이나 느낀 점을 기술하시오.**

테트리스 프로젝트를 진행하면서 실제로 게임이 진행되는 다이나믹한 프로그램을 짜볼 수 있어 흥미로웠다. 강의 자료의 flow chart가 꽤나 자세히 나와있어서 많은 도움이 되었고 이러한 긴 프로젝트에서 flow chart를 어떻게 구성하는 것이 좋은지 알 수 있었다. recommend기능을 구현할 때는 어떤 방법이 더 효율적일지 고민하면서 알고리즘의 중요성에 대해 깨닫게 되었다. 단순히 구현 방법이 떠오르면 코딩에 들어가던 때와 달리 알고리즘의 효율성을 따져보는 과정을 거치면서 앞으로의 프로그래밍 방향에 대해 고민하는 시간을 가졌다. 마지막으로 다양한 자료구조를 배우고 실제로 이용해볼 수 있었다.

**(끝)**