10주차 결과보고서

전공: 아트&테크놀로지학과 학년: 4학년 학번: 20191048 이름: 김도솔

**1.**

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

Tree 자료구조를 이용해 추천 시스템을 구현했다. Tree는 RecNode라는 구조체를 하나의 노드로 가지며 이 구조체를 통해 트리의 각 노드에 필요한 정보와 자식 노드들을 저장할 수 있다.

구조체 멤버 설명:

* level: 현재 노드의 레벨을 나타낸다. 트리의 최상위 루트 노드의 레벨은 0이며, 하위 레벨로 내려갈수록 값이 증가한다.
* accumulatedScore: 현재 노드까지의 누적 점수를 나타낸다. 각 노드는 자신까지의 점수를 가지고 있으며, 이 점수는 트리에서 경로의 점수를 계산하는 데 사용된다.
* recField: 현재 노드에서의 게임 필드 상태를 나타낸다. 2차원 배열로 표현되며, 필드의 높이와 너비에 해당하는 크기를 가지고 있다.
* child: 현재 노드의 자식 노드들을 저장하는 배열이다. 배열의 크기는 CHILDREN\_MAX로 정의되어 있으며, 자식 노드들에 대한 포인터를 가지고 있다.

*(작성 코드)*

typedef struct \_RecNode{

// Must-have element

int level;

int accumulatedScore;

char recField[HEIGHT][WIDTH];

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

} RecNode;

modified\_recommend 함수는 RecNode 구조체(root Node)를 인자로 받는다. 이 구조체는 게임 필드의 상태, 현재 레벨, 누적 점수 등을 저장한다. 함수 실행 후 각 노드들은 연결되어 tree 구조를 형성한다.

1. 각 블록의 회전 수를 결정하기 위해 nextBlock 배열을 참조하여 rotateNum을 설정한다.
2. 각 블록의 회전 수에 대해 필드 상에 놓일 수 있는 위치를 고려한다. x 값은 -2부터 WIDTH + 3까지 반복하며, 유효한 위치인지 확인한다.
3. 만약 블록이 이동 가능한 위치가 아니라면, 다음 위치로 넘어간다.
4. 현재 위치에 대한 child 노드를 생성하고, 블록의 회전 상태와 위치를 업데이트한다.
5. child 노드에 현재까지의 누적 점수를 저장한다.
6. 블록을 필드에 추가하고, 라인 삭제를 수행하여 점수를 계산한다.
7. 만약 현재 고려하는 레벨이 최대 레벨보다 작다면, 재귀적으로 modified\_recommend 함수를 호출하여 누적 점수를 계산한다.
8. 현재 고려하는 레벨이 최대 레벨과 같다면, accumulatedScore에 현재 누적 점수를 저장한다.
9. 현재까지의 최대 누적 점수와 현재 얻을 수 있는 점수를 비교하여 최대 누적 점수를 갱신한다.
10. 반복문을 통해 모든 회전 상태와 위치에 대해 계산을 완료하면, 추천된 블록의 회전 수와 위치를 저장하고 최대 누적 점수를 반환한다.
11. 위의 알고리즘은 가능한 모든 블록의 위치와 회전 상태를 고려하여 가장 높은 점수를 얻을 수 있는 블록의 위치와 회전 수를 추천하는 것이다. 이를 통해 게임 플레이어는 최적의 움직임을 선택할 수 있다.

modified\_recommend 함수의 시간 복잡도는 O(rotateNum \* WIDTH \* HEIGHT)이고, 공간 복잡도는 O(노드 사이즈 \* CHILDREN\_MAX ^ 고려할 블록 수)이다.

**2.**

블록의 모양에 따라 회전 수를 다르게 설정해서 생성되는 child node의 개수를 줄인다는 점에서 공간 복잡도 개선이 이루어졌다. 또한 modified\_recommed 함수의 경우 max score 값을 가지는 블록의 x좌표, y좌표, 회전 수가 지속적으로 업데이트 되기 때문에 Tree의 모든 노드에 동적 할당을 하는 대신 함수 내에서만 노드를 지역 변수로 선언해 메모리 사용 공간을 줄였다. 아쉬운 점은 pruning tree와 같은 기법을 사용하지 않아 극적으로 효율성이 개선되진 않았다는 점이다.

**3.**

Linked list, Tree, Array 등 각종 자료구조를 실제 알고리즘에 어떻게 활용할지에 대해 많이 배운 것 같다. 자료구조 수업 때 이론적으로 배운 개념들을 컴실 실습에서 직접 코드를 구현하며 실제 적용에 대해 알게되었다. Segmentation fault 같은 오류를 디버깅하고 수정하며 각 노드들이 어떤 구조로 연결되어 있는지, 어떤 식으로 변수들을 사용해야 하는지 등을 파악하는 과정이 힘들기도 했지만 재미있었다. printw(), scanw(), echo(), noecho(), getch() 등 ncurse 라이브러리의 다양한 함수들의 사용법에 대해 알게된 점도 좋았다.