10주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1.**

.................

실습과 과제에서 구현하는데 이용한 자료구조는 tree를 이용하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위와 같은 구조체로 노드를 정의하였다. tree의 level, 각각의 score, 그리고 각각의 field와 블록의 상태, parent와 child 노드의 주소를 저장하도록 구현했다. tree의 자료구조는 initTetris()에서 init\_Tree()를 통해서 tree를 초기화하면서 해당하는 레벨만큼의 tree를 구성하도록 하였다. ini\_Tree()에서는 root 노드에 대해서는 원래 field의 배열을 받아오고 level은 0, parent는 NULL 그리고 자식 노드의 개수는 1로 하고 나머지는 0으로 초기화한다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

child 노드는 VISIBLE\_BLOCKS만큼의 level만큼 생성하고, 각각을 초기화하면서 parent가 root로 연결되도록 하고, 재귀적으로 노드를 생성하도록 했다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

make\_Tree는 recommend 함수 내에서 Tree를 생성하도록 하는 함수이다. root 노드일 때는 nextBlock의 배열을 저장해두고, field의 배열을 노드 내에 업데이트한다.

텍스트, 스크린샷이(가) 표시된 사진

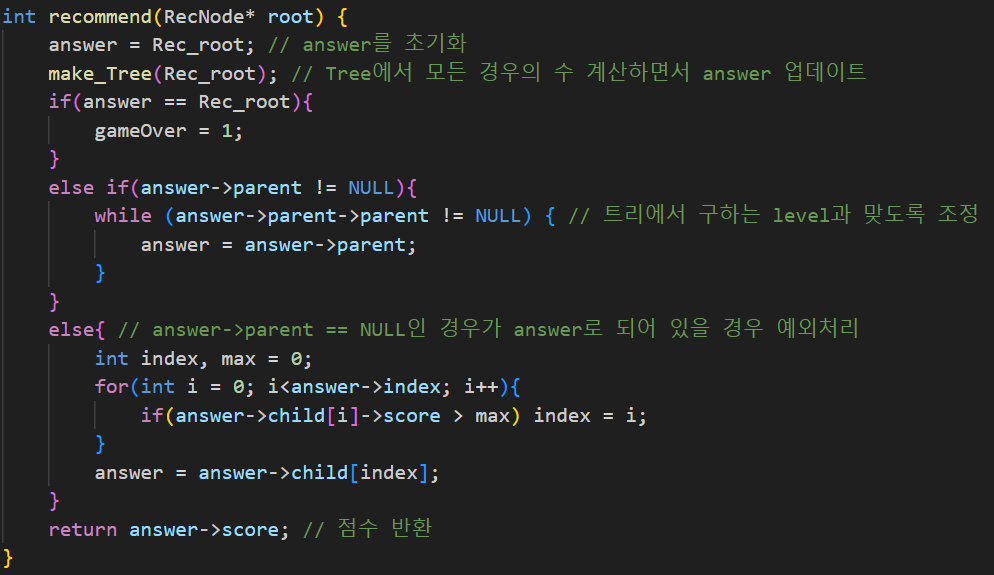
자동 생성된 설명

이 후는 공통적으로 모든 회전과 모든 x 좌표에 대해서 판별한다. 제일 위 쪽의 y좌표에서 CheckToMove 함수가 true일 때 자식 노드를 만든다. 자식 노드는 for문에 따라 회전수와 x 좌표를 넣어주고, 해당 위치에서 가장 아래로 내려간 y좌표를 구한다. child 노드의 score에는 field에 더할 때의 점수와 줄이 사라지는 점수를 더해주었다. 실습에는 점수로만 구현하였으나 개선하면서 y좌표의 제곱을 더해주는 과정을 추가했다. 이는 점수를 높이기 위해 한 쪽에 쌓아두는 과정을 줄이도록 선택한 방식이다. 그 후 정답 노드보다 점수가 높고 VISIBLE\_BLOCK-1 이상의 레벨의 노드인 경우에 answer노드를 해당 노드로 바꿔준다. 그 후 idx를 증가한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

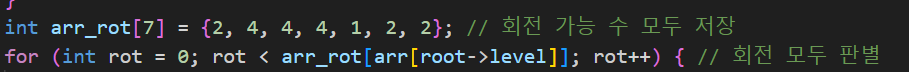
자동 생성된 설명

모든 과정 탐색 종료 후에는 root의 index에 idx를 넣어주어 자식 노드의 개수를 저장하도록 했고, 재귀적으로 호출하여 하위 노드도 모두 반복하도록 한다.

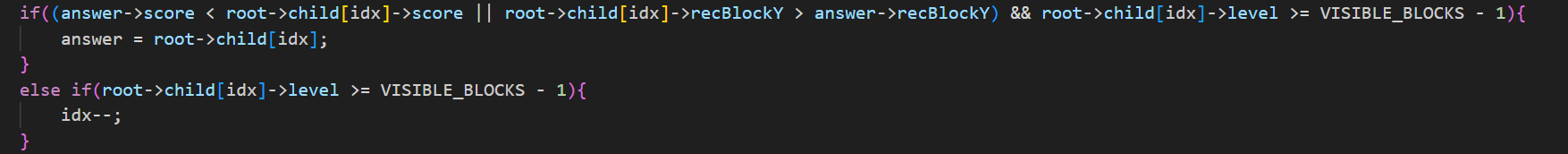


recommend 함수에서는 make\_Tree 함수를 호출하고, answer의 값에 따라 달라진다. answer가 Rec\_root이면 추천 블록이 없는 것이므로 gameOver를 하고, answer->parent가 NULL이 아닌 경우는 answer->parent->parent != NULL일 때까지 parent 노드로 이동해 가장 높은 레벨의 블록을 출력할 수 있도록 한다. 또한 예외처리로 answer->parent == NULL인 경우에 대해서는 하위 노드에 찾는 부분을 넣어두었다. 그 후 answer->score를 리턴하며 함수가 종료된다.

개선하기 전의 코드에서는 시간 복잡도가 O(CHILDREN\_MAX^VISIBLE\_BLOCKS)의 시간 복잡도를 가지고 공간 복잡도 역시 마찬가지이다.



개선한 경우에는 블록들의 회전 수를 배열로 정의하여 해당 배열에서 회전수를 찾아 그만큼 탐색하도록 개선하였다.



또한 y좌표를 비교해서 answer에 넣는 경우도 추가하였고, VISIBLE\_BLOCKS-1 이상의 레벨에서는 answer가 업데이트 되지 않은 경우에 idx를 1 줄여서 저장하지 않도록 구현했다. 그러나 시간 복잡도나 공간 복잡도가 바뀔 정도의 변화는 없어서 시간 복잡도와 공간 복잡도는 개선하기 전과 같다.

................

**2.**

.......................

모든 경우를 비교하는 트리와 비교했을 때 회전 수를 배열로 정의하여 정확히 필요한회전수만큼만 분석하도록 하여 탐색 시간을 줄이는 방식을 이용했다. 또한 answer에 업데이트 되지 않는 노드는 일정 이상 레벨인 경우에만 idx를 줄여서 저장하지 않는 방식을 사용하였다.

그러나 노드를 잘라내는 pruning이나 데이터의 단순화를 사용하지 않아서 시간 복잡도나 공간 복잡도에 있어서 큰 변화가 있을 정도의 구현을 하지 못한 점에는 아쉬움이 있다.

.........................

**3.**

.......................

평소에 잘 사용해보지 않았던 tree 구조에 대해서 많이 사용하며 연습해볼 수 있는 기회가 되었다. 특히 구조를 생성하고 접근하는 과정에서 메모리 오류를 고치는데 있어서 다양한 방법을 사용할 수 있었다. 또한 재귀를 사용하면서 발생하는 오류를 고치는데 gdb로 확인하는 과정을 거치면서 gdb를 사용한 디버깅에 좀 더 익숙해질 수 있었다. 또한 이번에 구현하는데 이용하진 않았지만 pruning을 통해서 좀 더 효율적인 접근을 해볼 수 있다는 점에 대해서도 새롭게 알 수 있었다. 또한 기존에 스켈레톤 코드로 주어진 play 함수 부분에 대해서도 recommendPlay를 구현하는 과정에서 참고하여 구현하면서 볼 수 있어서 어떻게 조금씩 떨어지는 것을 구현하는지에 대해서도 학습해볼 수 있었다.

.........................