**8주차 Tetris-3 예비보고서**

전공: 수학과 학년: 4 학번: 20161255 이름: 장원태

1. **교재를 참조하여 테트리스 프로젝트 3주차에 구현하는 추천 기능은 어떤 원리로 작동되는지 설명하시오. 그리고 추천 기능을 구현하는 tree 구조의 장점(효율성)과 단점(비효율성)을 기술하시오.**

이번 추천 기능을 구현하는 데에 이용할 자료구조는 트리(tree)이다. 트리 구조를 이용하여, 현재 블록과 다음 두 개의 블록 각각의 ID, Rotate, 위치, 그리고 필드 상태와 같은 정보를 토대로 각 블록이 놓일 수 있는 모든 경우의 수를 살펴보는 것이다. 그 모든 경우의 수 중, 가장 높은 점수를 받을 수 있는 경우를 찾아, 그 경우를 만들어 줄 수 있게 추천해주는 원리이다. 이 때, 현재 블록만으로 점수를 계산하는 것이 아니라 next block 두 개까지 모두 고려해야 한다는 점을 명심해야 한다.

이와 같은 트리 구조로 추천 기능을 구현하면 장단점이 있다. 우선, 장점으로는 하나의 시점에 생길 여러 경우의 수를 모두 고려할 수 있다는 장점이 있다. 이전 레포트에서 언급한 자료구조인 배열과 링크드 리스트(linked list)의 경우, 두 entity 간에 일대일로밖에 대응될 수 없다. 반면에, 트리를 이용하면 하나의 entity에서 여러 개의 entity로 일대다 대응이 가능하다. 이러한 부분에서, 블록 세 개의 배치에 대한 모든 경우의 수를 고려해야 하는 추천 시스템에서 트리 구조가 적합하다.

하지만, “모든” 경우의 수를 고려해야 한다는 점에서 트리 구조는 시간 및 공간 효율성이 떨어진다. 하나의 블록만을 고려했을 때, 그 블록에 대해 고려해야 할 경우의 수는 최대 34개까지 증가한다. 다음 블록도 최악의 경우인 경우 똑같이 34개, 그 다음 블록도 최악의 경우인 경우 34개가 된다. 이 모든 경우의 수에 대해 독립적으로 모두 구해야 하므로, 단순히 계산만 해보아도 최악의 경우 343=39304개의 경우의 수가 나온다. 즉, 트리에 39304개의 노드가 생성된다는 의미이다. 뿐만 아니라, 각 노드는 구조체 형태로 구성되어 있어, 하나의 데이터만을 저장하는 것도 아니다. 따라서, 더 많은 블록을 고려해야 할수록 시간 및 공간 복잡도가 기하급수적으로 증가한다는 문제가 있다.

1. **Tree 구조의 비효율성을 해결할 방법에 대해서 2가지 이상 생각하고, 그 idea를 기술하시오.**

우선, 위와 같이 트리 구조가 비효율적인 이유는 고려할 경우가 너무 많기 때문이다. 그리고, 이는 블록을 더 많이 고려할수록 기하급수적으로 늘어난다. 그래서, 애초에 블록을 많이 고려하지 않는 것이 좋다. Next block를 단 하나만 놓고, 현재 블록과 그 다음 블록 하나만 고려하더라도 최대 342=1156개의 경우의 수만 고려하면 되며, 이는 꽤 혁신적으로 효율을 높여준다. 그러나, 그만큼 점수를 높게 받는 플레이를 추천한다는 추천 기능의 취지에는 더 안 좋고, 그에 따라 더 점수를 높게 받을 기회를 놓칠 수도 있다.

다음으로, 수업 자료에 있던 pruning tree를 적용하는 것이다. 즉, 하나의 노드의 가지(branch) 중에서 특정 점수 이상 주지 못하는 트리는 잘라버리는 형태이다. 예를 들어, 블록 중 단 한 면만이 바닥에 맞닿아 나오는 “10점”보다는 더 높도록 기준을 설정하는 것이다. 이렇게만 해도 불필요한 경우의 수를 줄일 수는 있다. 그러나, “줄이 지워지는 경우”에 발생하는 점수를 고려하지 않은 것이며, 10점이 나온 경우에서 그 후에 줄이 지워져 더 높은 점수가 나올 수도 있는 가능성이 있어 이들을 모두 고려하는 것도 힘들 수 있다.

위의 pruning tree와 비슷한 맥락으로, 한 depth에서 경우의 수를 탐색하며 각 경우에 대해 deleteLine 함수를 적용해보는 것이다. 이 때, 해당 함수는 줄이 지워지거나 지워지지 않는 경우에 대한 점수를 리턴하게 된다. 만약 해당 함수가 0보다 큰 값을 리턴하는 경우가 생기면, 이전까지 고려했던, 해당 함수값이 0인 가지들을 잘라버리는 것이다. Pruning tree에서 조금 변형한 생각이지만, 이 경우 역시 잘라버린 가지에서도 그 다음에 deleteLine 함수값이 0보다 크게 되어 점수가 더 높게 나올 수 있는 가능성이 있기 때문에, 이까지 모두 고려하는 것이 어려울 것이다.