8주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

**1. 추천 기능 구현**

이번에 구현하는 추천 기능은, 트리 구조를 이용한다. 현재 블록이 필드에 놓여지는 모든 경우의 수 중 하나의 경우를 하나의 노드로 하여 트리를 구성한다. 이때, 놓여지는 블록의 점수를 계산하여, 각 노드에 저장한다. 그런 다음, 각 노드에 대해서 다음블록이 놓여지는 경우의 수 만큼 자식노드를 생성하고 트리를 만든다. 마찬가지로, 이때 블록이 놓여졌을 때 얻어지는 점수는 누적되어 저장된다. 생성되는 트리의 깊이는 보여지고 있는 블록의 수 만큼 생성된다. 따라서, 이 테트리스의 경우 깊이가 3인 트리가 생성되게 된다.

노드에 저장된 점수를 비교하여 가장 높은 점수를 저장하고 있는 노드를 택하면, 테트리스 게임을 진행함에 있어 합리적인 경우를 선택할 수 있게 된다. 택한 노드의 x좌표 y좌표 그리고 회전 수를 입력받고 필드에 반영하여 표시한다.

이 구조의 장점은, 깊이가 깊어질수록 더 이상적인 경우를 선택해 추천이 가능하다는 점이다. 즉, 미래에 결정할 수 있는 정보량이 많아질수록 추천기능은 더욱 정교하고 정확하게 된다.

하지만, 최선의 경우를 선택하기 위해서 모든 경우의 수를 탐색한다는 점에 있어서 깊이가 깊어질수록 탐색해야하는 경우의 수는 기하급수적으로 증가하게 된다. 탐색해야하는 노드의 수가 많아진 다는 의미는 시간복잡도와 공간복잡도의 효율성이 떨어진다는 것과 동치다. 또한, x좌표들에서 얻어지는 점수가 같을 경우 x좌표가 가장 왼쪽인 경우를 선택한다는 점에서 이상적인 수가 아닐 수 있다.

**2. Tree 구조 개선 방법**

트리의 구조의 시간 효율성을 개선하기 위한 방법으로는 대표적으로 pruning기법, 가지치기 기법이 있다. 현재, 탐색해야하는 노드의 가지수가 너무 많기 때문에 모든 경우의 수를 탐색하기에는 현실적으로 어렵다. 따라서, 현재 블록을 놓았을 때 얻어지는 점수를 비교하여 최고점수를 가지는 노드만 살리고 나머지 노드에 대해서는 다음 블록이 놓이는 경우를 확인하지 않는 것이다. 이후, 다음 깊이로 넘어가 같은 작업을 반복시행한다.

예를 들면, 현재 블록이 필드에 놓이는 경우는 총 n 이라 가정하자. 이때 생성되는 노드의 가지수는 n 임을 알 수 있다. 다음 블록의 경우도 마찬가지로 k개라 한다면, 완전탐색의 경우 총 경우는 n\*k 임을 알 수 있다. 하지만, 첫 번째 시행에서 가장 높은 점수를 가지는 1가지만을 확인한다면, 총 확인하는 경우는 k로 시간적인 측면에서 효율적이라 할 수 있다. 하지만, 이러한 경우 최고의 경우를 항상 선택하기는 힘들다. 이 경우 점수가 높은 노드를 몇 개 추출하여 경우를 확인하는 방법도 존재한다.

앞서 말한 트리 구조는 공간적인 측면에서도 비효율적이다. 앞서 말한 가지치기 방법에서 더 나아가, 사용하지 않는 노드의 메모리를 해제함으로서 공간을 낭비하지 않게된다. 더욱 나아가, 각 노드는 비어 있는 필요 없는 필드의 정보도 저장하고 있다. 이에, 각 줄의 높이 정보만을 저장하여, 공간의 효율성을 증대시킬 수 있다. 이때, 비어있는 부분이 채워져 있다고 판단 할 수도 있기 때문에 추천기능의 알고리즘이 더 좋지 못할 가능성도 존재한다.

각 블록이 놓아졌을 때 곱해지는 가중치를 이용한 알고리즘도 존재한다. 예를 들어, 블록이 놓였을 때 블록의 최고 높이가 높을수록 음의 가중치를 주거나, 지워지는 줄의 수가 많다면 양의 가중치를, 빈칸이 생기는 개수에 따른 가중치 등의 방법이 존재한다. 이 경우 다음블록과는 상관없이 현재 상황에서 놓이는 경우 중 최선을 선택할 수 있다. 단순하게 점수로 판단하지 않는 점에서 최대한 사람과 비슷한 생각을 하는 추천기능을 구현 할 가능성이 증가한다.

구멍이 나는 부분에 대해서 전혀 신경을 쓰지 않는 경우는 다음과 같다. 만약 놓아지는 블록 아래로 생기는 빈칸의 수에 대해 예로 -10의 가중치를 주게 되면 아래 그림처럼 놓았을 경우 계산되는 점수는 -10\*2 + 필드와 닿는 면적 2\*10 = 0 이다. 또한, 일자블록을 세로로 놓게 된다면, 블록의 높이가 높아지므로 역시 음의 가중치를 줄 수 있다. 따라서, 선택해야하는 가중치는 경험적으로 반복시행을 통해 수정해나가야 함을 알 수 있다. 다음과 같은 상황에서 블록의 최대높이가 더 높은 것과 빈칸이 생기는 상황 등을 고려할 수 있다.

텍스트, 점수판이(가) 표시된 사진

자동 생성된 설명