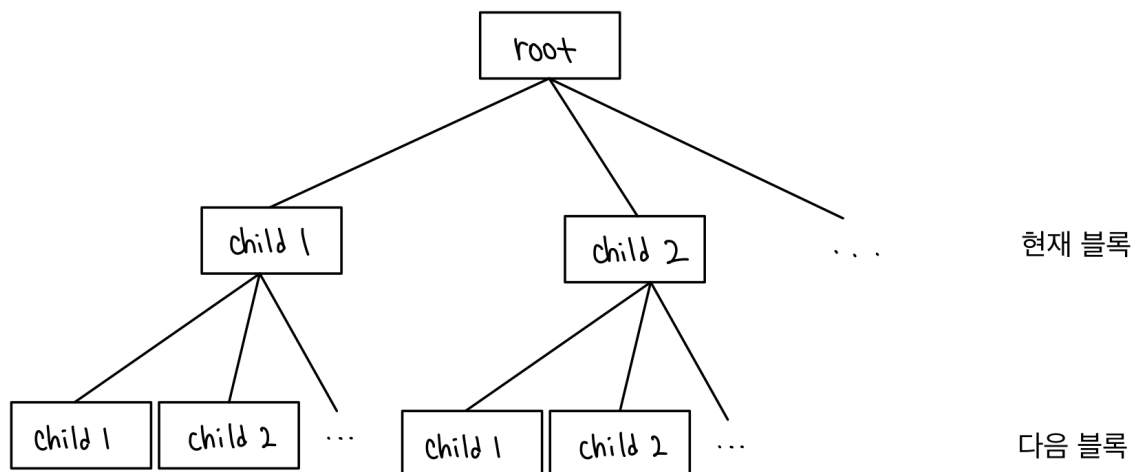


1. 테트리스 프로젝트 3주차에 구현하는 추천 기능이 작동되는 원리

테트리스 추천 시스템은 tree(트리)를 이용해 구현한다. 각 노드는 블록 모양 및 회전수, 블록 위치, 점수, 필드 상태 등으로 구성된다. 아래 그림과 같이 현재 블록이 어떤 모양을 가지며 어떤 회전수를 가지고 어떤 위치에 놓일 수 있는지 경우의 수를 파악한 뒤, 각각에 대해 점수를 계산한다. 각각의 상황을 자식 노드로 만들어준다. 지금까지 구현한 테트리스 게임은 다음 블록의 정보도 알고 있기 때문에 현재 블록이 해당 위치에 놓였을 경우 다음 블록이 놓일 수 있는 경우의 수도 계산하여 각 상황에 대해 점수 계산까지 가능하다. 이 역시 자식 노드의 자식 노드로 만들어준다. 마찬가지로 다음 다음 블록의 정보까지 확인할 수도 있다. 트리에서 가장 아래의 자식 노드까지 누적 점수를 가장 크게 만드는 길을 따라 해당 자식 노드들의 블록 모양 및 위치를 추천해주는 시스템이다.



2. 추천 기능을 구현하는 tree 구조의 장단점

(1) 장점(효율성)

미래의 블록을 알고 있는 경우 모든 경우의 수를 고려한다면 현재 상황에서 큰 score를 만드는 최선의 선택을 할 수 있다.

(2) 단점(비효율성)

미래의 경우의 수를 많이 고려할수록 시간과 공간의 소비가 매우 커진다. 블록의 모양에 따라 블록이 놓일 수 있는 좌표 수, 회전수 4가지를 고려하면 최악의 경우 34가지가 가능하고, 현재 블록의 child, 다음 블록의 child, 다음 다음 블록의 child를 고려하게 된다면 노드가 $34^1 + 34^2 + 34^3$ 개이다. 만약 다음 n개의 블록을 고려한다면 $\sum_{k=0}^n 34^k = \frac{34^{n+1}-1}{33}$ 개로, 기하급수적으로 증가한다. 또한 이렇게 개수가 많은 각 노드에서 필드 정보를 저장하려면 공간 소모 역시 기하급수적으로 증가한다.

3. Tree 구조의 단점(비효율성)을 해결할 방법에 대해 2가지 이상 생각하고, 그 idea에 대해 기술하시오.

(1) Pruning tree: 시간 및 공간의 비효율성 개선


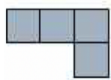

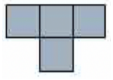

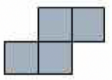
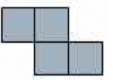
트리의 각 level에서 모든 경우의 수를 노드로 표현하지 않고, 각 level에서 가장 높은 점수를 가지는 child 노드만 남기고 나머지 노드들은 삭제하는 방법이 있다. 이 방법을 이용한다면 삭제된 노드들의 자식 노드들을 구할 필요가 없으므로 시간 및 공간 소모가 감소한다. 하지만 삭제된 경로 중에서 그 다음 노드까지 고려할 때 가장 큰 누적 점수를 갖게 되는 경우가 발생할 수 있다. 따라서 남겨 놓는 노드의 수를 k개로 정할 수 있다. k의 크기에 따라 알고리즘의 효율성이 달라질 것이다.

(2) Data simplification: 공간의 비효율성 개선

트리의 각 노드들은 필드의 정보를 저장하는 데 많은 공간을 사용한다. 이를 줄이기 위해서 필드의 높이만을 고려해 저장하는 방법이 있다. 이 경우 10x22의 공간 대신 10(너비)의 공간이 필요하므로 사용하는 메모리 공간을 줄일 수 있다.

(3) 블록 모양에 따른 회전수 개별적 처리

테트리스에서 사용되는 블록은 총 7가지인데, 다음과 같이 블록에 따라 회전수에 차이가 있다.

shape ID	0	1	2	3	4	5	6
블록 모양							
회전수	2	4	4	4	1	2	2

회전하여도 모양이 같은 블록의 경우 모양이 달라지는 경우에 대해서만 고려해준다면 트리에서 노드의 개수를 줄일 수 있다. 이는 tree 구조의 단점을 개선할 수 있다.