8주차 예비보고서

전공: 아트앤테크놀로지학과 학년: 3학년 학번: 20201116 이름: 이수빈

**­­1.**

테트리스 게임을 할 때 현재 블록을 어떠한 rotation으로 어떠한 위치에 놓을지에 따라 각기 다른 결과를 가져온다. 만약, 앞으로 n개의 테트리스 블록 모양을 미리 안다면, 모든 경우의 수를 다 계산해보면 최종 하나의 최선을 선택을 할 수 있다. 이번 3주차 실험에서는 실제로 nextblock의 정보를 갖고 모든 경우의 수를 고려하여 최종적으로 가장 고득점을 가져다 줄 블록의 rotation과 위치를 추천하는 시스템을 구현해야 한다.

이 시스템을 구현하기 위해서는 여러가지 정보를 담고 있는 노드로 이루어진 tree를 사용한다.

Graphical user interface, text

Description automatically generated

각 노드는 위와 같은 정보를 담고 있는데 필수적으로 각 노드의 depth (lv), 누적된 점수 (score), field의 정보 (\*f[WIDTH]), 그리고 child node를 가리키는 pointer (\*c[CHILDREN\_MAX])로 이루어진다. 알고리즘을 나타내면 아래와 같다.

for i <- 1 to 4 //블록의 회전 수에 대해서

for w <- 0 to w //필드상 놓일 수 있는 위치에 대하여

child 노드 정의

if(tmp->lv < BLOCK\_NUM+1) recommend (cur);

recommend 블록 설정

장점

모든 경우에 수에 대해 다 살펴 볼 수 있기 때문에 항상 최선의 해를 제공한다.

단점

1. 시간의 비효율성

모든 경우의 수에 대하여 n개의 블록을 고려하게 된다면, 34^n 개의 경우의 수가 생기므로 기하급수적으로 시간 소요가 늘어난다. 계산 시간의 증가로 프로그램이 느려질 수도 있다.

2. 공간의 비효율성

한 개의 노드가 차지 하는 공간의 크기를 c라 할 때, 총 필요한 공간의 크기는 c\*34^n이다. 이는 시간 복잡도와 마찬가지로 기하급수적으로 늘어나기 때문에 overflow가 일어날 수도 있다.

**2.**

1. 시간의 비효율성 해결 방법 – Pruning tree

각 노드마다 형성되는 34개의 child의 노드 중, 최고의 성능을 내지 못할 branch는 없애 버리는 방식이다. 다만, 이를 고려할 때에는 해당 branch가 미래에 어떠한 성능을 가져올지 예상 해야함으로 주의해야한다.

2. 공간의 비효율성 해결 방법 – Data Simplification

모든 노드마다 공통된 부분을 제외하고 저장하거나 필요한 데이터만 추출해서 저장하면 된다. 필드를 저장할 때 이미 쌓여져 있는 블록의 정보도 저장이 되는데 이는 누적으로 되기 때문에 각 부모 노드가 갖고 있는 필드 정보와 child노드의 필드 정보는 거의 똑같다. 따라서 이를 이용하면 누적되는 정보만 저장하는 방식으로 공간을 아낄 수 있다. 또 다른 방법으로는 필드의 높이만 저장하는 것이다. 필드에 블록을 쌓을 때, 가장 높이 있는 블록만 고려 된다. 만약 n번째 줄에 블록이 있다고 가정하였을 때, 1~n-1 번째 줄과 상관없이 그 다음 블록은 n+1부터 쌓을 수 있다. 이점을 이용하여 가장 위에 위치한 블록의 높이만 저장하면 전체 필드를 저장하지 않아도 되어 공간을 훨씬 절약할 수 있다.