10주차 예비보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

자료구조 중 Tree를 사용하여 구현한다. 현재 필드에 대한 정보를 담은 루트 노드(level 1)를 생성 한 다음, 현재 테트리스 블록(nextBlock[0])이 놓일 수 있는 모든 경우의 수(회전수\*위치)만큼의 자식 노드를 생성한 다음, 각각 해당 경우에 블록을 놓았을 때의 상황을 기록한다. 각각의 자식 노드에서는 다음 테트리스 블록(nextBlock[1])이 놓일 수 있는 모든 경우의 수를 계산한다. 같은 방식으로 level이 VISIBLE\_BLOCKS에 도달할 때까지 자식 노드를 생성하면서 모든 경우의수를 계산한다. 가장 높은 점수를 얻을 수 있는 경우를 선택하여 추천 할 블록의 위치와 방향(회전상태)를 결정한다. 블록이 한 번 쌓인 다음부터는 이미 생성된 노드의 정보를 갱신하는 방식으로 재사용하여 게임이 끝날 때까지 추천 기능이 수행된다.

**2.**

2-1. 장점

: 모든 경우의 수를 탐색하므로 주어진 상황에서 탐색할 깊이 내에서는 반드시 최적의 선택을 할 수 있다. 또한, 단순한 Tree 구조를 사용함으로써, 프로그램의 구현이 쉽다.

2-2. 단점 :

한 테트리스 블록을 놓을 수 있는 가짓수는 최대 34가지이므로 순수하게 일반적인 Tree만으로 추천기능을 구현했을 때, 추천 함수의 시간 및 공간복잡도는 모두 최소 O(34^level)을 가지게 되므로 성능을 올리기 위하여 탐색 깊이를 늘렸을 때, 기하급수적으로 소요시간이 증가하거나, 메모리 초과의 가능성이 커지게 된다.

**3.**

3-1. Pruning Tree :

자식 노드중에서 누적된 점수가 상위 n개에 해당하는 노드만 남기고 나머지는 모두 쳐내고 계속 진행하는 방법이다. 이 경우, 시간 및 공간복잡도가 최소 O(34^level)에서 O(n^level)로 줄어들게 되어 프로그램의 성능이 개선된다. (n < 36)

3-2. Data Simplication :

tree의 노드 중 가장 큰 데이터를 차지하는 것은 char recField[HEIGHT][WIDTH]인데,

총 220바이트가 필요하지만, 하나의 원소는 0 또는 1의 값만을 가지므로

220bits (=28bytes 필요)만을 사용해 해당 변수의 크기를 줄일 수 있다.

3-3. 불가능한 경우의 생략

추천 위치를 계산할 때, 현재 블록을 놓는 모든 경우에서 더 이상 필드에 놓을 경우가 존재하지 않을 때(=GameOver) 해당 노드는 더이상 계산하지 않고 생략함으로써 시간복잡도를 줄인다.

3-4. 점수 반영비의 다각화

추천 위치를 결정하는 점수를 단순하게 실제 점수와 같은 방식으로 계산하지 않고,

1. pruning tree 적용시 설정할 기준 (블록이 쌓인 높이, 블록이 쌓이는 y좌표, 지운 줄의 개수)

2. 추천위치 결정시 계산할 점수의 기준 (기존 점수 계산법 + 옆면에 닿은 블록 및 벽의 개수)

위와 같은 방식으로 낮은 깊이로 탐색하더라도, 더 효율적으로 높은 점수를 얻을 수 있도록 추천위치 설정시 점수 반영을 다각화 한다.