10주차 결과보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

1. 실습 시간에 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술하시오.

이번 실습 시간에는 recNode(즉 recommend 시스템을 구현하기 위해서 트리 자료구조를 사용했다.

우선, 헤더 파일에 트리의 구조를 설정한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

RecNode로, 트리의 깊이(처리하는 경우의 수), 점수, 그리고 자식 노드를 선언하고, 그에 해당하는 필드, 그리고 추천할 위치인 reBlockX, reBlockY 그리고 추천할 recBlockRotate 회전수로 구성했다.

우선 테트리스 추천 시스템의 recommend 함수를 구성하기 위해서 트리를 구성해야 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

분석할 블록의 어떤 회전수와 어떤 위치를 하나의 child node로 받는다. CHILDREN\_MAX는 36으로 나올 수 있는 최대의 경우의 수이다. child는 \_RecNode를 리스트로 갖는 데이터이기 때문에, child를 CHILDREN\_MAX로 초기화하고, 이 각 index의 child의 lv, score, field를 초기화한다. 또한, 분석할 블록(NextBLock)의 개수를 3개 ,VISIBLE\_BLOCKS로 설정했다. 그 이유는 4가 넘어가는 순간에 테트리스 게임의 의미가 무색해질 만큼 느린 속도로 게임이 진행되기 때문이다. tree를 3 level로 만든다.

2. 완성한 알고리즘의 시간 및 공간 복잡도를 보이시오

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위 그림 3개는 recommend 함수를 나타낸 것이다. 간단하게, 이 프로그램의 알고리즘을 설명하자면, 블록의 회전수인 최대 4가지의 경우에 대해서 반복을 하면서 각 경우에 대하여 필드에 위치할 수 있는 지 없는 지 경우의 수를 파악한다. 가로의 길이만 생각한다면,  이 블록은 가로의 길이가 3이다.  하지만 회전시킨 이 블록은 가로의 길이가 2이다. 가로의 길이가 다르기 때문에 필드의 위치하게 되는 경우의 수가 다르다. 이를 (757~791) 줄까지 파악한다. 그 이후는 필드를 child node에 저장하고, y좌표를 다른 블록이 닿을 때까지나 바닥에 닿을 때까지(더 이상 움직이지 않을 때)까지 증가시킨다. 그 후 child의 score를 갱신한다. 어떤 임의의 위치와, 임의의 회전수를 갖고 있기 때문에 score를 갱신하고, 만약 더 고려해야 할 블록이 있다면 (lv < VISIBLE\_BLOCKS – 1)이라면 재귀적으로 호출한다. 그 후 추천할 위치와 회전수를 초기화하고, max 점수를 return한다.

이 함수의 반복문과 재귀함수가 고루 쓰였기에 이를 확인하면 된다. 반복문의 블록의 WIDTH, 블록의 HEIGHT , ROTATE, 즉 모두 상수인 4이다. 그 후 재귀 호출은 VISIBLE\_BLOCKS만큼 실행된다. 이 한 블록을 고려하는 최대의 경우의 수는 34이므로, 따라서 시간 복잡도는 O(34^VISIBLE\_BLOCKS)고 볼 수 있다.

이 함수의 공간 복잡도는 함수의 RecNode의 구조체의 자료 구조를 살펴보면 알 수 있다. 입력 크기에 따라 메모리 사용량이 변하지 않기 때문에 공간 복잡도는 상수이다.

실습 시간에 구현한 모든 경우의 수를 파악하는 함수는 level 하나에서 최대의 경우의 수가 34이므로, 현재의 블록 1개와 다음 블록 2개인 블록 3개를 고려한다면 34의 3제곱에 해당한다. 즉 정확도를 높이기 위해서는 여러 블록을 고려해야 하지만 시간 복잡도가 너무 큰 폭으로 증가하게 된다. 강의 자료에서 소개한 pruning tree를 사용한다면 이를 줄일 수 있다.

사실 예측 프로그램은 현재 나의 상태로 상상도 못한 알고리즘이었다. 하지만, 자료구조 수업 때 배운 내용과 강의 자료에 나와 있는 내용을 천천히 생각하면서 코드를 짜니 구현이 가능했다. 예측 프로그램이란, 많은 경우의 수를 고려해야 정확도가 올라가기 때문에, 이 시간 복잡도를 줄이며 데이터를 읽는 방식, pruning tree 뿐 아니라 다른 알고리즘들도 공부해야겠다고 생각했다. 매 함수나, 반복문을 쓸 때에도 시간 복잡도를 신경 쓰며 코딩을 할 때면 불편하지만, 결과물이 효과적으로 돌아가는 것을 필수적인 과정인 것 같다.