8주차 TETRIS3 결과 보고서

전공 : 수학 학년 : 4 학번 : 20151378 이름 : 진상우

1. 실험시간에 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술하시오. 완성한 알고리즘의 시간 및 공간 복잡도를 보이시오.

- 자료구조

앉아있는, 검은색, 하얀색, 표지판이(가) 표시된 사진

자동 생성된 설명

모든 경우를 고려하려고 tree를 이용하여 c를 구성하였다.

- 알고리즘

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

처음엔 input으로 들어온 정보를 옮겨 저장하고, CheckToMove를 이용하여 현재 위치에서 표현할 수 있는 가장 아래까지 내린다. 그러고 새로운 필드에 해당 자리에 블럭을 그린뒤, 바닥에 닿는 면적을 touch로 저장하였다. 그러고 해당 필드에서 DeleteLine함수를 사용하여 삭제할 수 있는 라인지 존재하는지 보고, cs에 삭제한 점수를 기록한다. 새롭게 가상으로 만든 곳에서 recommend함수의 recursive를 이용하여 다음 블럭이 쌓였을때 점수를 추가한다. Recursive는 두번까지 가능하며, 최대 가상 점수보다 새로운 점수가 크거나 같으면 더 낮게 블럭을 놓을 수 있을 때, 새롭게 갱신하고 현재의 recommend가 최초로 호출되었다면 해당 블럭의 좌표를 recommendX와 recommendY에 해당되는 블럭의 회전 정보를 recommendR에 저장한다. 해당 모양이 x축을 타면 이를 반복하고, 회전하는 경우에 대해서도 반복한다.

이에 대한 시간복잡도는 O((WIDTH\*HEIGHT)^3), 공간복잡도는 O((WIDTH\*HEIGHT)^3)으로 같다.

1. 모든 경우를 고려하는 tree구조와 비교해서 어떤 점이 더 향상되고, 어떤 점이 그렇지 않은지 아울러 기술하시오

모든 경우를 고려하는 tree구조는 모든 경우의 수를 다 따져보아야 하기에 공간, 시간이 비효율적으로 사용된다. Tree의 깊이가 깊어질 수록 엄청나게 경우의 수가 증가하기 때문이다. 이를 해결하기 위해서 블럭의 개수를 줄이면 된다. 매우 뒤의 블럭을 고려하지 않고, 최근의 블럭만 고민하면되는데 그치만 이러면 예측할 수 있는 경우가 작아져 최적의 점수는 아니다. 또 다른 방법으로는 branch를 잘라내는 pruning tree를 이용하는 것으로, 최대의 점수가 되지 못하는 경우를 미리 제외시켜 필요하지 않은 탐색을 줄이는 것이다. 이를 사용하면 시간복잡도를 줄일 수는 있지만 효과적인 점수를 내기 힘들 수 있기에 pruning하는 적절한 방법이 필요한 단점이 있다. 마지막으로는 데이터를 간단하게 받아 자료구조를 간단하게 하는 방법이 있다. 현재는 각 노드별 이전 노드의 필드 정보들을 모두 받아오는데, 필드에서 전체가 아닌 각 열의 높이로 받으면 이차원배열이 아닌 일차원배열로 만들 수 있기에 데이터의 양이 줄어 공간복잡도가 매우 줄 수 있다. 하지만 열의 높이만 받기에 효율적으로 아래의 줄을 제거하지 못해 최적의 점수를 내기엔 부족할 수 있을 것이다.

1. 본 테트리스 프로젝트를 통해 습득한 내용이나 느낀점을 기술하시오.

이번 테트리스 프로젝트를 통하여 tree에 관해 더 자세히 생각해보고 알 수 있게 된것 같다. 자료구조에서 배웠던 DFS알고리즘을 이용해 최적의 방법을 알 수 있었고, 34가지에 달하는 다중트리를 다루면서는 어떤 위치에서 함수가 필요한지 볼 수도 있었다. 또한 방대한 양의 코드를 이용하면서 알고리즘을 더 면밀히 생각하면서 신중히 공부해볼 수 있었던 것 같다. 어디서 잘못하였는지 알기도 쉽지 않아 각각의 경우를 따지는 것이 어려웠지만 이러한 프로그램을 작성해보면서 색다를 경험을 하게 되었던 것이 좋았다.