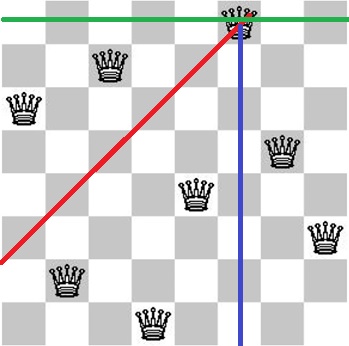
인공지능 레포트

Assignment #1

|  |  |
| --- | --- |
| 과목명 | 인공지능 |
| 과제명 | N-queens Problem |
| 학과 | 컴퓨터 전공 |
| 학번 | 2011004028 |
| 이름 | 이성훈 |

1. 문제 정의

N-queens problem은 N by N 행의 행렬에서 queen N개를 배치하는데 가로, 세로, 대각선 방향으로 queen이 서로를 공격할 수 없게 배치하는 것이 핵심이다.



(N이 8일때의 예시)

이 문제를 DFS, BFS, DFID의 방법으로 풀어보려고 한다.

1. 코드 분석

코드는 DFS, BFS, DFID로 탐색하는 코드와, 상태 체크하는 코드 나뉘어 진다.

|  |
| --- |
| 자료구조 선언 |
| * ArrayList라는 자료구조를 사용했고, ArrayList의 각 항들의 순서는 N queens 문제의 행을 뜻한다. 그리고 각 항들의 실제 값은 열을 뜻합니다. 예를 들어 A라는 ArrayList가 <3, 1> 이라면 첫 번째 행 4번째 열, 2번째 행 2번째 열에 퀸이 있는 상태를 나타낸다. 그리고 이 ArrayList를 트리의 노드로 사용한다. * DFS, DFID 에서는 STACK을 사용하고 BFS에서는 QUEUE를 사용했다. |
| DFS |
| * 코드는 처음에 depth 1인 노드를 삽입하는 것으로 시작한다. 눈여겨봐야 할 점은 빨간색으로 표시된 부분처럼 N-1번째 노드를 먼저 삽입해서 트리의 왼쪽부터 search 하도록 했다. (Stack의 자료구조는 후입선출이기 때문이다.) * Stack에 들어간 후로는 DFS알고리즘을 통해 탐색한다. * 출력형식을 맞추기 위해 goal의 state를 String s에 저장했다. |
| BFS |
| * DFS와 BFS의 가장 큰 차이는 자료구조를 BFS에서는 Queue를 선택했다는 것이다. 트리의 왼쪽부터 Search하기 위해 DFS와 다르게 for문의 순서를 변경할 필요는 없었다. (Queue는 선입선출이기 때문이다.) |
| DFID |
| * DFID는 내부 구조는 동일한데 다만 depth\_limit이라는 int형 변수를 정의해서 depth 1부터 하나씩 늘려서 iterative하게 깊이를 늘려가는 알고리즘이다. * DFS와 마찬가지로 트리의 왼쪽부터 search하기 위해 반복문을 노드 삽입 순서를 N-1부터 0까지로 정했다. |
| 상태 체크 |
| * 이번 과제의 핵심 코드라고 할 수 있는 check 함수이다. * 각 노드를 input으로 넣으면 output의 결과는 해당 노드가 N-Queens 규칙에 위배되지 않는지를 확인한 후 위배 되면 False, 괜찮으면 True를 출력하는 함수이다. * N-Queens 규칙은 가로, 세로, 대각선에 하나의 퀸만이 위치 할 수 있음이다. |

1. 실행화면

|  |
| --- |
| Jar file을 명령 프롬프트 상에서 실행시켰을 때 파일 출력 |
| C:\Users\qweqa\AppData\Local\Microsoft\Windows\INetCacheContent.Word\실행화면.jpg   * 출력결과는 같은데 그 이유는 Tree의 왼쪽부터 Search를 하게 모든 알고리즘을 구성했기 때문이다.      * 3을 input으로 넣었을 때의 No solution 출력 상황 |