**인공지능**

2019062833 컴퓨터 소프트웨어학부 김유진

**1. 함수 설명**

1) bfs(N): 퀸의 개수 N을 입력 받아서 큐 자료구조를 이용한 bfs 알고리즘으로 해를 구한다. 노드의 길이가 N이되면 잡아먹을 수 있는 퀸의 페어의 개수를 구하고 0이라면 해, 아니라면 탐색을 계속한다.

2) hc(N): 퀸의 개수 N을 입력 받아서 해를 찾을 때까지 반복해서 랜덤 시작을 한다. Local 탐색이므로 별도의 자료구조는 사용하지 않으며 현재의 상태에서 모든 이동에 대해 잡아먹는 퀸의 페어의 수를 구한 후 가장 적은 방향으로 이동한다. 만약 잡아먹는 퀸 pair의 개수가 0이 된다면 해답이며 어디로도 이동할 수 없는 경우는 Stuck이므로 다시 재시작 한다.

3) csp(N,depth,node): 퀸의 개수 N, 현재 dfs에서의 depth, 현재 퀸들의 위치가 기록된 배열인 node를 받아온다. 이미 해가 갱신되어 있다면 더 이상 진행하지 않고 그렇지 않다면 해를 찾는다. depth가 N이 되면 해가 찾아진 것이며 그렇지 않다면 back tracking을 계속 진행한다.

4) attack pair(node,N): node, 즉 각 N개의 퀸들의 row 위치가 기록되어 있는 배열을 받아서 서로 공격하는 퀸의 pair가 몇개인지 구하여서 리턴하는 함수. 우선 같은 row에 있는 지 검사한 후 대각선 방향에 같이 놓여있는지를 확인한다.

5) write\_file()/read\_file(): 파일에서 문제를 읽어 들이며, 해답을 output 파일에 쓰는 함수들

**2. 알고리즘 설명**

1) BFS

- 넓이 우선 탐색 기법으로 자료구조 큐를 이용한 방식이다.

- 이미 탐색한 곳은 큐에 넣지 않으며, 큐에 들어가 있더라도 이미 탐색되었다면 확장하지 않는 것이 규칙이다.

- 우선 큐 안에 시작 지점을 넣고 확장하게 되며 먼저 들어간 곳을 먼저 확장하게 되고, 확장된 것들은 큐의 뒤로 들어가서 대기하게 된다.

- 가까운 곳부터 탐색하게 되므로 Optimality가 보장된다

2) Hill-Climbing

- Local 탐색으로, 이전의 상태를 저장하기 위해 스택이나 큐 등의 자료구조를 사용하지 않아 메모리를 거의 사용하지 않는다.

- 경로를 구하는 문제에는 적합하지 않고 최소/최대와 같은 최적의 해를 찾는 문제에 적합하다.

- 현재 상태에서 값이 좋아지는 방향으로 이동하게 되므로 당장에 좋은 것 만을 고르게 되어 Optimality를 보장하지 않는다.

- 그렇기 때문에 랜덤한 재시작을 통해 어느정도 Optimality를 보장할 수 있다.

- 랜덤하게 배치를 설정한 뒤 현재 상태에서 이동가능한 움직임에 대해 휴리스틱 함수 값을 모두 구한 뒤 가장 바람직한 방향으로 이동하게 된다.

- 만약 해가 나오지 않은 상태로 더 바람직한 움직임이 불가능 해진다면 stuck 상태가 되어 재시작이 필요하다.

3) Constraint Satisfaction Problems

- 다른 문제와 다른 점은 제약이 걸려있다는 점이다. N-퀸 문제의 경우 서로 잡아먹을 수 있는 퀸의 페어가 0이 되는 것이 제약 조건이다.

- 탐색 방법 중 하나로 변수는 고정한 채로 할당만을 고려하는 백트래킹 방식이 있으며, DFS를 이용하여 구현한다.

- 먼저 첫 변수에 대한 할당을 부여하고 그 다음 변수, 다음 변수로 내려가면서 해 공간을 탐색하게 되는데 제약 조건에 어긋나게 되면 백트래킹으로 되돌아 간다.

- 모든 변수의 할당이 제약조건을 만족하게 된다면 해를 구한 것이다.

- 백트래킹 방식 이외에 포워드 checking 방식이 있으나 이 과제에서는 사용하지 않았다.

**3. 실험결과: N=5에 대한 각 알고리즘 별 output 캡쳐**

텍스트이(가) 표시된 사진

자동 생성된 설명