Assignment1

컴퓨터소프트웨어학부

2017029952 여현규

1. 함수 설명
   1. 2017029952\_assignment\_1.py

* main()

input.txt를 open하고 내용을 한 줄씩 읽는다. 숫자 n과 n-queen을 푸는데 사용할 algorithm에 따라 각 함수 bfs, hc, csp를 호출한다.

* 1. bfs.py
* bfs(n)

output을 위한 파일을 생성하고, bfs를 통해 n-queen solution을 구하는 과정을 수핸한다. 결과를 output 파일에 저장한다.

* goal(arr, num)

입력받은 arr이 n-queen의 조건을 만족하는지 평가하고, 만족하면 True를 return한다.

* 1. hc.py
* hc(n)

output을 위한 파일을 생성한다. hc를 통해 n-queen solution을 구하는 과정을 수행한다. 결과를 output 파일에 저장한다.

* h(n)

hill climbing을 수행할 때, n개의 column끼리 비교하여, 최소 점수(제일 적게 공격하는 경우)를 가지는 column을 찾아 최소 점수와 해당 column 그리고, 바뀔 row를 return 해준다.

* calculate\_score(arr, col, num)

col인자로 들어온 한 개의 column 내부 row들 중에서 현재 본인의 row를 제외하고, row를 바꿀 경우, 최소 점수를 가지는 row를 구한다.

* get\_score(arr, num)

들어온 arr이, 몇 개의 서로 공격하는 queen pair를 갖는지 구해 return한다.

* 1. csp.py
* csp(n)

ouput file을 만들고, dfs를 실행한다. Backtracking 결과를 받아 출력한다.

* go\_next\_level(arr, cur\_col, num)

종료 조건을 검사한다. 종료 조건이 아니라면, dfs로 n-queen arr을 한 column씩 재귀함수로 결정해 나가는 역할을 한다.

* location\_test(arr, cur\_col, num)

n-queen 배열을 검사하여, constraint에 걸리는지, 걸리지 않는지 test후 조건을 만족한다면 True, 만족하지 않다면 False를 return한다.

1. 알고리즘 설명

모든 경우에 arr[]에서 index는 column을 의미하고, arr[index]는 해당 column에 놓인 queen의 row값을 의미한다. 2차원 배열 형태를 1차원 배열 형태로 비교한 것이다. 한 Column내에 두개의 queen이 올 수 없다는 성질을 이용했다.

2-1. bfs algorithm

bfs이기 때문에 queue를 사용한다. 초기에 비어 있는 arr을 queue에 집어넣어준다.

만약, queue가 비어 있다면, 후보로 test할 arr이 전부 소진된 것이기 때문에 답이 없는 경우를 의미한다.

텍스트이(가) 표시된 사진

자동 생성된 설명queue pop()으로 선택된 배열 중에서 goal check의 대상은 num \* num board여야한다. arr의 길이가 num보다 작다면, 즉, len(test) < num이면, (num – i) \* (num) 배열을 의미한다. 따라서, column을 추가하여, 다시 queue에 넣는 과정을 거친다.

Column을 추가할 때, row값을 0~num-1까지 차례로 추가시켜 queue에 넣음으로 bfs의 형태를 갖게 된다. 만약, queue에서 pop한 arr이 num\* num이라면, 즉, len(test) == num이면, 우선, 배열의 크기가 n\* n이므로 goal check의 대상이 된다. goal함수에서 true를 return한다면, 정답이므로 file에 출력하고, 종료한다.

텍스트이(가) 표시된 사진

자동 생성된 설명Goal을 확인하는 과정은 왼쪽 코드와 같다. 순차적으로 arr의 왼쪽 값(i)을 선택한다. i보다 오른쪽에 있는 값(j)를 검사한다.

arr[i] == arr[j]이면 같은 row에 있는 것을 의미한다. arr[i] – arr[j] == i - j라면 i가 j의 왼쪽 위에 있음을 의미한다. arr[i] – arr[j] == j - i라면, i가 j의 왼쪽 아래에 있음을 의미한다.

위의 제약에 하나라도 걸리면, goal이 될 수 없음으로 False를 return 한다. 모든 제약에 걸리지 않는다면, goal이므로 True를 return 한다.

정리하자면, bfs에서 한 개의 column씩 가능한 모든 row의 경우를 생성하고, child로 넘어갈 때도, 가능한 row들을 모두 붙여준다. 이러면, bfs를 이용한 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명2-2. hc algorithm

우선 output파일을 생성한다. Hill climbing에서 최초의 arr[0~num-1]은 0 ~ num-1에서 random한 값으로 생성한다. 또한, 이전 score와 바꾸고 난 후의 즉, 현재 score를 비교하기 위해 이전 score를 저장하는 변수 pre\_score를 선언한다. pre\_score를 정수의 최댓값으로 설정해주는데, 거의 대부분의 n-queen에서 공격 최댓값을 같는 경우가 없을 것으로 생각해서 이다.

답을 구하지 못해, random으로 재성성해 hc를 진행하는데, 너무 많이 재시작할 수 있기 때문에 해당 부분을 통제하기 위해 count 변수를 선언했다.

While문은 답을 찾거나, 찾을 수 없을 경우에 break를 한다. h(arr, num)함수를 통해, 현재 arr에서 최소 점수, 해당 col, 최소 점수를 가능하게 하는 row를 return 받는다.

만약, hc 답을 구할 때, 1000번의 restart가 진행됐으면 찾는 과정을 멈춘다. 1000번의 값은 임의로 설정했으며, n에 따라서 답이 있는 경우도 있겠지만, restart를 1000번 진행하는 과정에서 모든 경우에 답이 없는 상태로 고착됨을 의미한다. 혹은, n에 따라 실제 답이 없는 경우도 1000번의 restart를 진행한 후 no solution이라는 결론에 도달한다.

Score가 0이라는 것은 col에 해당하는 column의 값을 h에서 return 받은 row로 바꾸면, solution이 됨을 의미한다. 따라서, 값을 바꿔주고, 정답을 출력한다.

Score가 0보다 클 때, 두가지 경우로 나눌 수 있다. 첫째, 이전 과정에서 구한 score(pre\_score)보다 현재 구한 score가 크거나 같다면, row를 바꾼다고 해도 효과가 없다. 따라서, restart를 위해 arr를 random으로 생성한다. Restart 횟수를 check하기 위해 count 값을 1 증가시킨다. 또한, pre\_score를 sys.maxsize로 초기화 해준다. 둘째, 이전 score보다 현재 구한 score가 작다면, 해당 row로 바꿨을 경우 정답에 가까워짐을 의미한다. 따라서, pre\_score를 현재 score로 바꿔주고 arr도 현재 row값으로 바꿔준다.

텍스트이(가) 표시된 사진

자동 생성된 설명hc(n)함수는 n개의 column끼리 가지고 있는 최소 점수를 비교한 후 column과 해당 row, score를 return한다.

h(arr, num)함수는 for문 내부에서 calculate\_score(arr, col, num)을 호출하여 각 column내부에서 row를 0~num-1까지 바꿔보며 최소 score가 되는 row와 해당 score를 return 받는다. 따라서, 해당 값을 저장하기 위해 each\_score[]와 each\_row[]를 선언했다. 또한, 비교하기 위하여, min\_score와 min\_col을 선언했다. 만약, column들끼리 같은 최소 score를 가진다면, random을 통해 선택하도록 했다.

텍스트이(가) 표시된 사진

자동 생성된 설명Calculate\_score(arr, col, num)은 한column내부에서 row값을 바꿔갈 때 갖는 score를 계산한 후, 최소 score를 갖는 row를 return한다. 만약, row끼리 같은 score를 갖는다면, random을 통해 row를 선택한다. Score는 임의로 int max를 선언했다. 왜냐하면 구해질 score집단보다 큰 수를 선택해야지 오류의 확률이 작기 때문이다. 또한, 현재 위치와 같은 row일때는 score를 계산하지 않도록 했다. arr을 변경시킬 경우, python은 변수를 call by reference로 받기 때문에 임의의 배열 tem을 사용했다.

텍스트이(가) 표시된 사진

자동 생성된 설명Get\_score(arr, num)은 받은 arr이 총 몇 개의 공격 queen pair를 갖는지 계산해서 return 해준다.

정리하자면, hc에서 column들끼리 점수를 비교하기 위해 h를 호출한다. h는 각 column내부에서 최솟값을 갖는 row정보를 얻기 위해 calculate\_score를 호출한다. calculate\_score는 후보로 선택한 row들이 각각 어떤 score를 갖는지 알기 위해 get\_score를 호출한다.

hc: score가 0인지, 혹은, 개선의 여지가 있는지 파악

h: column들끼리 비교해서 최소 score를 갖는 column을 선택

calculate\_score: 한 column 내부에서 최소 score를 갖는 row를 선택

get\_score: 후보로 들어온 arr의 score를 계산.

2-3. csp algorithm

Csp는 dfs와 backtracking algorithm을 통해 구현했다. Column별로 작은 index이면 parent로 취급했다. Parent부터 조건을 따져가며 후보군을 만들고, child(next column)를 붙여 나가는 방식으로 진행했다.

우선 csp에서는 재귀함수를 멈추기 위해 전역변수 count와 재귀함수에서 빠져나올 때 column값이 감소할 경우, child node의 값들을 초기화 시켜줄 때 사용하는 전역변수 pre를 사용했다. 또한, 찾은 정답 arr를 옮겨주기 위한 전역list result[]를 사용했다.

텍스트이(가) 표시된 사진

자동 생성된 설명 Csp함수에서 output file을 만든다. Count를 0으로 설정하는데, 한개의 input file에서 csp가 여러 번 실행될 때, count를 다시 초기화 하기 위함이다. 또한, arr을 n의 크기만큼 만들고, 각 내부의 값을 int의 max값으로 설정한다. 이는 추후에 해당 위치를 test할 때, 아직 배정되지 않은 child node(next column)를 제외하고 계산하기 위함이다. 그후, go\_next\_level(arr, cur\_col, num)을 호출한다. 재귀적 함수 호출 종료 후, count가 0이라면 답이 없는 경우이고, count가 0보다 크다면 답이 있는 경우이다. 이는 go\_next\_level에서 결정된다.

텍스트이(가) 표시된 사진

자동 생성된 설명Go\_next\_level(arr, cur\_col, num)은 재귀적으로 호출된다. 우선, 재귀함수 호출을 더 이상 하지 않는 조건을 살펴보면, 현재 cur\_col값이 num이랑 같다면, 0 base이기 때문에 이전의 조건들 즉, n-queen board가 조건을 만족했다는 것을 의미한다. 따라서, count를 1늘려준다. Count가 1보다 크다는 것은 답을 찾고 더 찾기 위해서 추가로 진행함을 의미한다. 따라서, 해당 arr는 정답이 아니다. 또한, csp는 최초 한 개의 값만을 출력해야하기 때문에 다음과 같이 설정했다. Count가 1보다 크지 않다면, 즉 1이라면 정답이므로 답을 출result에 옮겨준다.

만약, cur\_col이 num과 같지 않다면, 아직까지 탐색해야할 child node가 있음을 의미한다. 따라서, 현재 가지고 있는 cur\_col들을 row를 바꿔가며 가능한 값이 있는지 파악해야 한다. 따라서, location\_test(arr, cur\_col)을 통과한다면, go\_next\_level을 재귀적으로 호출한다. 만약, 재귀적으로 빠져나왔을 때의 column값이 이전보다 작다면, 그 후 column을 초기화 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명 Location\_test(arr, cur\_col)은 주어진 arr이 사용가능한지 판단한다. 현재까지 후보군으로 정해진 cur\_col까지만 파악한다.

정리하자면, 재귀적으로 함수를 호출하여, 한 column씩 붙여 나간다. 해당 후보군이 실패한다면, parent의 값을 바꿔 다시 재귀적으로 파고든다. 이는 dfs backtracking을 이용한 방식이다.

1. 결과 출력

텍스트이(가) 표시된 사진

자동 생성된 설명

N = 5일 때, 결과는 다음과 같다. hc는 첫 arr이 random하기 때문에 실행시마다 답이 바뀔 수 있다.