CS 480 Artificial Intelligence Project 1 Fall 2024

Due: September 16, 2024

The projects will be done by two students per team. Please submit your source code through canvas. Each team will submit just one program. Please follow instructions regarding submission given at the end.

In this project, you will implement a recursive version of Depth-First Search (DFS) to solve the following problem: Given a placement of queens in a checker-board so that none of them attack any other, your program shall place as many additional queens in the remaining squares as possible so that none of the queens (original ones as well as the added ones) attack each other. This is a problem that has been extensively implemented using different algorithms, most frequently using DFS. The variant we consider is that the board is cyclindrical in that the first and the last columns are considered to be next to each other. This will change the rules for attack along the diagonal. For example, in a standard 8 by 8 board, the queens in (row, column) = (0, 0) and (row, column) = (1, 7) are not attacking each other. But in a cyclindrical board they do. For this problem, there is no need to keep the explored set since the graph has no cycles and no multiple paths to reach the same node.

The input to your program will be a vector B of length n (for the $n \times n$ version of the problem such that $B[i] = j$ if there is a queen placed in position (row, column) = (i, j). $B[i]$ = -1 if there is no queen is in row i. The output will be a list v of length n with the same convention. Note that all the queens in the input shall be retained and have extra queens (as many as possible) replacing -1 values.

To get you started, a python program **queenDFS.py** has been provided which solves the problem of queen placement in the standard (planar) board. It takes as input a vector B in which the first k = length of B rows have a queen placed. A pre-condition is that these k queens do not attack each other. A second parameter n is the number of rows and columns of the board. $B[j]$ represents the column number in which the queen is placed in row j. It returns a vector by placing $n - k$ additional queens and returns a board with n queens, if such addition is possible. (But this program will not find the correct solution when n queens can't be placed on the board.) This program works with a normal flat board. You are to make the following changes: (i) make it work for a board in which the left and right boundaries of the board are glued to form a cyclinder shaped board and, (ii) modify the program so that it returns a solution with the largest number of additional queens that can be placed in a given partially filled board, (iii) allow the input to skip rows (for example, the input may have queens in rows 1, 3 and 4) and (iv) report the total number of nodes expanded by DFS.

```
Algorithm:
Input: A board B in which the k-th member is -1 (indicating no queen in row k or a nonne
Pre-condition: No two queens in the list attack each other.
Output:  board B in which as many additional queens have been added to the input list su

Algorithm c_queensDFS(B):
if no additional queens can be added to B:
    return 1, B
count = 0
best = copy of B
for each cell c in B:
   if c does not contain a queen and c is not under attack by queen in B:
          place a queen in cell c of B
          count1, temp = amazonDFS(B)
          count = count + count1
          if temp has more queens than best:
               best = copy of temp
    remove queen from cell c
return count+1, best
```

Implementation Details: When your main program runs, it asks the user to enter a board as a list of n numbers. The user shall enter -1 for rows containing no queens. It returns a board by adding as many additional queens as possible, and the total number of nodes expanded by DFS.

Some test cases are provided soon (in the updated version).