
course

Announcements (announcements)

About the Course (https://swayam.gov.in/nd1_noc20_cs53/preview)  Ask a Question (forum)

Progress (student/home)  Mentor (student/mentor)

Week 12 Programming Assignment 2

Due on 2020-04-23, 23:59 IST
8 queens problem. Successively finds all positions with queens in the first i columns. Given a position with i queens, the next one needs to be checked for intersection with these i.

A chess board has 64 squares in 8 rows and 8 columns. Queens placed at position (r,c) (i.e. row r and column c) and (r', c') are said to capture each other if r=r' or c=c' or |r-r'|=|c-c'|. A puzzle asks, find a configuration of 8 queens on distinct squares such that no pair of queens capture each other.

In this problem you will write the main part of the program to find all possible configurations of n queens on an nxn chessboard. The idea is as follows. We will progressively construct a vector confQueens, where confQueens[i] will eventually contain the set of capture free configurations with i queens in the first i columns. A single configuration is simply a vector of i numbers, where the ith number gives the row position of the queen in the ith column. We start by setting confQueens[0] to the empty vector representing the empty board. Then in general, given the set of configurations confQueens[i], we create confQueens[i+1] as follows. For every configuration pos in confQueens[i], we create new configurations by adding a queen to pos. Of these, those that are capture free are added to confQueens[i+1]. This addition is to be done by the function appendnext. You have to write this function, while the main program is given.

The function appendnext has two arguments: confqueens[i+1] which is the set of capture free configurations with i+1 queens found so far, pos, which is a configuration with i queens & 'n', the size of the board (nxn). We must construct configurations pos||0, pos||1, ..., pos||n-1 where pos||j denotes the configuration obtained from pos by adding a queen in column i+1, row j. Note that pos is just a vector and this appending operation can be done using the push_back operation. Of these configurations those which are capture free should be added to confQueens[i+1]. Note that we only need to check whether the i+1th queen captures the previous ones, because we already know that the first i do not capture each other.

Note that if you specify the board size to be 1 (input to the program), you will get a single 0 as output (configuration). This is because in a board of size 1x1, you can place a queen at the single available position i.e. in the first column (in the first row). In a board of size 4 (4x4) there are 2 configurations where 4 queens can be placed without capture. For a input of 4, the output should be:

```
1302
2031
```

```c++
int main(){
    int n; cin >> n; // boardsize
    vector<set<vector<int>>> confQueens(n+1);
    // confQueens[i] = configurations with i queens, i=0..8
    confQueens[0].insert(vector<int>()); // empty board
    for(int i=0; i<n; i++){
        // generate successive confQueens
        for(auto pos : confQueens[i])
            appendnext(confQueens[i+1], pos, n);
    }
    for(auto conf : confQueens[n]){ // Print all configurations.
        for(auto q : conf)cout << q; // They will be printed in increasing
        cout << endl; // order because they are in a set.
    }
}
```
Sample Test Cases

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>135024 251403 304152 420531</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1302   2031</td>
</tr>
</tbody>
</table>

The due date for submitting this assignment has passed. As per our records you have not submitted this assignment.

Sample solutions (Provided by instructor)

```cpp
#include <iostream>
#include <cmath>
#include <map>
#include <vector>
#include <set>
#include <string>
using namespace std;

#define repeat(x) for(int _iterator_i = 0, _iterator_limit = x; _iterator_i < _iterator_limit; _iterator_i++)
#define main_program int main()

int main(){
    int n; cin >> n; // boardsize
    vector<set<vector<int>>> confQueens(n+1);
    // confQueens[i] = configurations with i queens, j
    confQueens[0].insert(vector<int>()); // empty board
    for(int i=0; i<n; i++){
        // generate successive conf(}
        for(auto pos : confQueens[i])
            appendnext(confQueens[i+1], pos, n);
    }
    for(auto conf : confQueens[n]){ // Print all configurations.
        for(auto q : conf)cout << q; // They will be printed in increasing order because they are in a set.
        cout << endl;
    }
}
```