Analysis for task "Mirrored game"

CodeIT, 2014-2015, Round 6

The first thing we see is that in order to be able to move all black chips in one board there no two black chips can be in mirrored positions.

If this is fulfilled then we have to count the number of black chips in both boards. Then we will move all chips from the board which contains less black chips to the other board and the required number of moves is equal to the number of chips on the board we are moving from.

To illustrate the solution I will use a fragment from the author's source code (C++):

1. // size of the boards
2. int n;
3. // mat[row][col] - two-dimensional array with chips
4. // the first board is: rows = 0 ... n-1
5. // the second board is: rows = n ... 2n-1
6. // 0 = white chip, 1 = black chip
7. int mat[2 \* MAX\_N][MAX\_N];
8. ...
9. void solve ()
10. {
11. // the number of black chips in each board
12. int countBlack[2] = {0, 0};
13. // iterate over each cell from the first board
14. for (int row = 0; row < n; row++)
15. {
16. for (int col = 0; col < n; col++)
17. {
18. // find the mirrored row
19. int invRow = 2\*n - row - 1;
20. // check if we have two black chips in mirrored positions
21. // if so we print -1 and leave the function
22. if (mat[row][col] && mat[invRow][col])
23. {
24. printf("-1\n");
25. return;
26. }
27. // update the number of black chips in each board
28. // in the array mat: 0 = white chip, 1 = black chip
29. countBlack[0] += mat[row][col];
30. countBlack[1] += mat[invRow][col];
31. }
32. }
33. // if everything went OK during the iteration
    * 1. // the we have the count of the black chips on each board
34. // and we print the smaller of the two counts
35. printf("%d\n", std::min(countBlack[0], countBlack[1]));
36. }
37. Author: Nikola Stoyanov