SCHOOL OF COMPUTING
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

Course Code: 18CSC204J

Course Name: Design And Analysis Of Algorithm

Title of the Experiment	Finding defective chessboard by using divide and conquer.
Name of the candidate	GAJULAPALLI NAGA VYSHNAVI
Team members	Shruthi Kannan (RA2011003010037) Aryan Sinha (RA2011003010066)
Register Number	RA2011003010049
Date of the experiment	27-06-2022

Staff Signature with date

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

S.R.M NAGAR, KATTANKULATHUR – 603 203

BONAFIDECERTIFICATE

REGISTER NO.____________________

Certified to be bonafide record of the work done by _________________ of_____ B.tech degree course in the practical

18CSC204J- Design and Analysis of Algorithms in SRM Institute of Science and Technology, Kattankulathur during the academic year 2021-22.

Date:

Lab Incharge:

Submitted for university examination held in

SRM Institute of Science and Technology, Kattankulathur.

TABLE OF CONTENTS

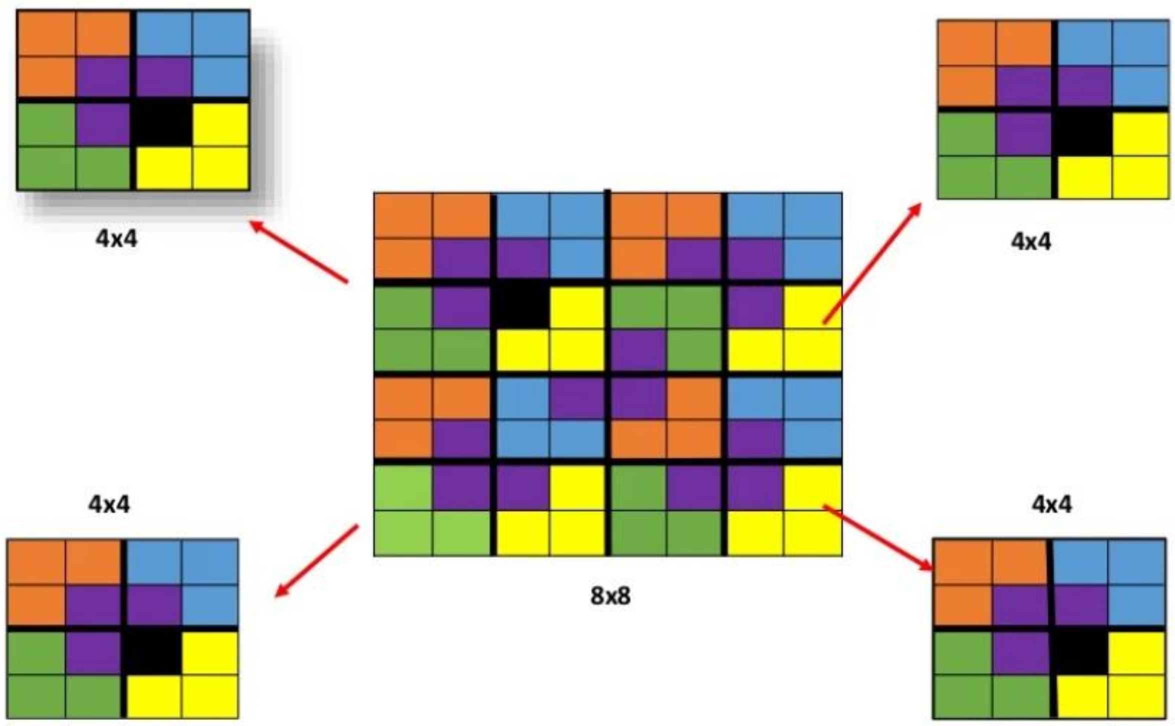
Abstract:

The defective chessboard problem is solved with the help of divide and conquer approach.So in this defective chessboard problem of size nxn the board has one missing cell of size 1x1.Task is to fill the board using L-shaped tiles. L-shaped tile is a 2x2 square which has one of the cells it’s dimension of 1x1 missing. Task is to fill the board using L-shaped tiles such that it covers the entire chessboard except the missing cell.

SECTION 1:

PROBLEM STATEMENT:

Finding defective chessboard by using divide and conquer.



DESCRIPTION:

Selection of Algorithm Design Strategy

DIVIDE AND CONQUER

Divide and conquer is an algorithm design paradigm. A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. This technique can be divided into the following three parts:

1. **Divide:** This involves dividing the problem into smaller sub-problems.
2. **Conquer:** Solve sub-problems by calling recursively until solved.
3. **Combine:** Combine the sub-problems to get the final solution of the whole problem.

SECTION 2:

METHODOLOGY:

Steps to find the defective chessboard:

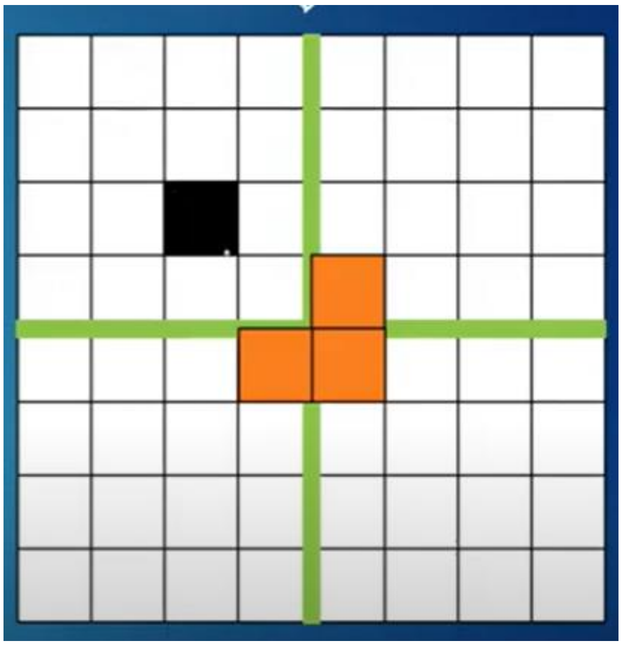
1. We have a chessboard of size n x n , where n =2k.Exactly on square is defective in the chessboard.The tiles are in L-shape i.e. 3 squares. Objective Cover all the chessboard with L-shape tiles, except the defective square.
2. It is possible to cover all non-defective squares. Let's see how. As the size of the chessboard is n x n and n=2k. Therefore, Total no. of squares =2k x2k =22k. No. of non-defective squares = 22k-1.Now, for the value of K,22k-1 is divisible by 3.
3. Defective chessboards. 1x1 2 x2 2 x2 2 x2 2 x2 As the Size of these chessboards displayed here, 1x1 and 2x2 we don't have to place any L-shape tile in the first and rest can be filled by just using 1 L-shaped tile. For 4x 4 chessboard.
4. For 8X8 DEFECTIVE CHESS BOARD.
    - One of the cell is defective.

- We divide the chess board into equal sub half's.Creation of defective box.
- Trick to cover the chess board with tiles.
- Again creation of defective boxes as we divide the chess board division of problems into sub-problems.
- As we have finally divided the problem into 2x2 board we will put the tiles.
- The procedure will continue until all the sub board are covered with the tiles.
- The final chess board covered with all the titles and only left with the defectives which we created.
- Here we will cover the defectives which we have created as in the last, there should be only one defective left.Combining of all sub problems.
- 8x8 4x4 4x4 4x4 4x4
- 2x2 2x2 2x2 4x4 2x2
- 1x1 2x2 4x4 8x8

SECTION 3:

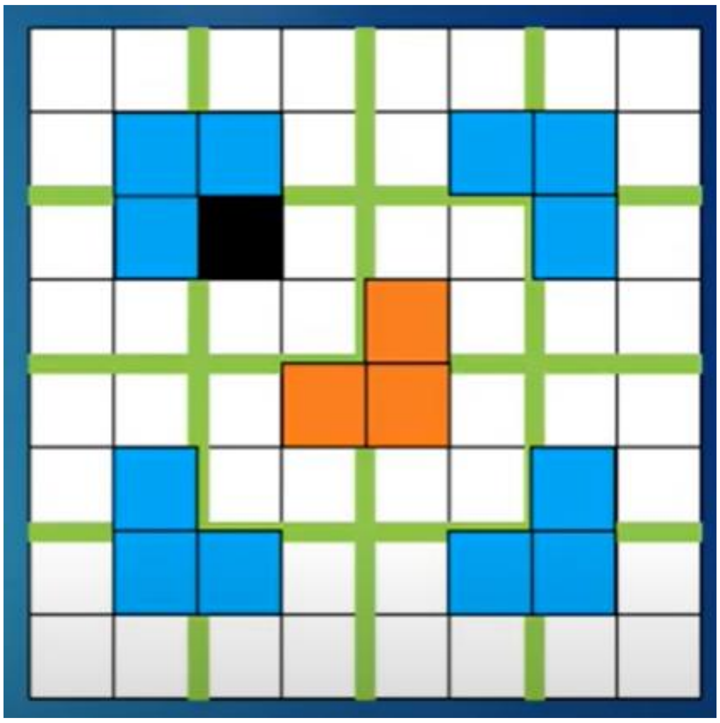
ALGORITHIM:

Step-1:
Apply Divide and conquer and divide the cell as 4x4 cells
Step-2:
Place a L shaped tile at the centre such that it does not cover the n/2*n/2 sub square that has a missing square.
Now all four sub-squares of size n/2*n/2 have a missing cell



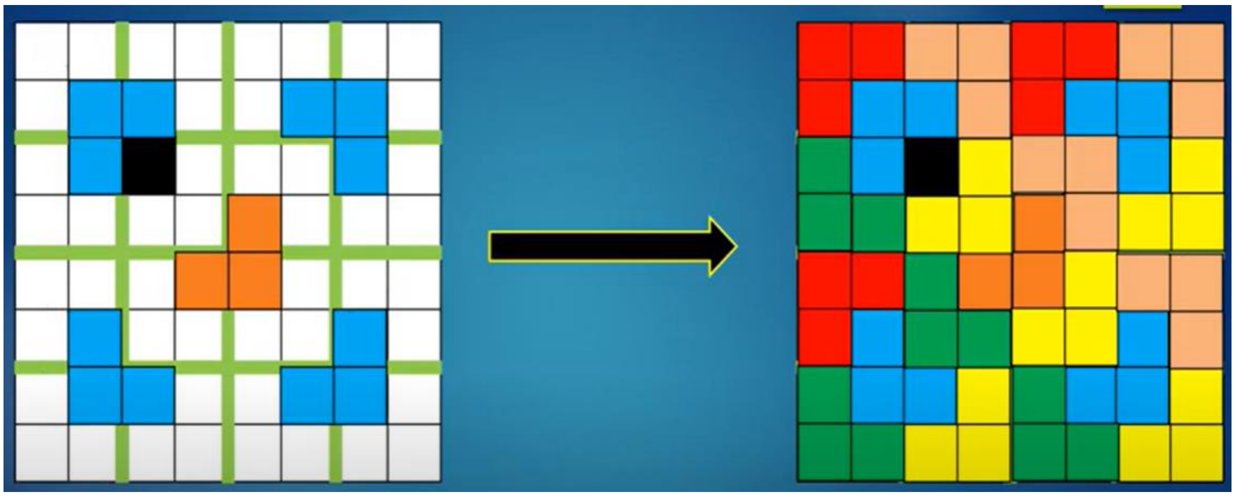
Step-3:

Now the 4x4 sub square is divided into further 2x2 squares (ie) solve the problem recursively until the base case is reached. To make every sub-problem a smaller version of our original problem add defects at the center, with one square in each of the four quadrants of our board, except for the one with our original defect would allow use to make four proper sub-problems, at the same time ensuring that we can solve the complete problem by adding a last trionimo to cover up the pseudo-defective squares we added.



Step-4:
Once the base case is reached then fill the L shaped tiles.
After solving each of the four sub-problems and putting them together to form a complete board, we have 4 defects in our board: the original one will lie in one of the quadrants, while the other three were those we had intentionally added to solved the problem using DAC. All we have to do is add a final trionimo to cover up those defects.
Every recursive algorithm must have a base case, to ensure that it terminates.So we have 2x2 block with a single defect as a base case.

Step-5:
Now the chess board except the missing tile part has been successfully been placed with L shape tiles



Code:

```
//MINI PROJECT-FINDING DEFECTIVE CHESSBOARD
//RA201103010049
//RA201103010037
//RA201103010066
#include <bits/stdc++.h>
using namespace std;
int size_of_grid, b, a, cnt = 0;
int arr[128][128];
// Placing tile at the given coordinates
void place(int x1, int y1, int x2,
           int y2, int x3, int y3)
{
    cnt++;
    arr[x1][y1] = cnt;
    arr[x2][y2] = cnt;
    arr[x3][y3] = cnt;
}
int tile(int n, int x, int y)
{
    int r, c;
    if (n == 2) {
        cnt++;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (arr[x + i][y + j] == 0) {
                    arr[x + i][y + j] = cnt;
                }
            }
        }
        return 0;
    }
    // finding vaccant location
    for (int i = x; i < x + n; i++) {
        for (int j = y; j < y + n; j++) {
            if (arr[i][j] != 0)
                r = i, c = j;
        }
    }
    // If missing tile is 1st quadrant
    if (r < x + n / 2 && c < y + n / 2)
        place(x + n / 2, y + (n / 2) - 1, x + n / 2,
              y + n / 2, x + n / 2 - 1, y + n / 2);
    // If missing Tile is in 3rd quadrant
    else if (r >= x + n / 2 && c < y + n / 2)
        place(x + (n / 2) - 1, y + (n / 2), x + (n / 2),
              y + n / 2, x + (n / 2) - 1, y + (n / 2) - 1);
    // If missing Tile is in 2nd quadrant
    else if (r < x + n / 2 && c >= y + n / 2)
        place(x + n / 2, y + (n / 2) - 1, x + n / 2,
              y + n / 2, x + n / 2 - 1, y + n / 2 - 1);
    // If missing Tile is in 4th quadrant
    else if (r >= x + n / 2 && c >= y + n / 2)
        place(x + (n / 2) - 1, y + (n / 2), x + (n / 2),
              y + (n / 2) - 1, x + (n / 2) - 1,
            y + (n / 2) - 1);

    // dividing it again in 4 quadrants
    tile(n / 2, x, y + n / 2);
    tile(n / 2, x, y);
    tile(n / 2, x + n / 2, y);
    tile(n / 2, x + n / 2, y + n / 2);
    return 0;
}
int main()
{
    size_of_grid = 8;
    memset(arr, 0, sizeof(arr));
    a = 0, b = 0;
    arr[a][b] = -1;
    tile(size_of_grid, 0, 0);
    for (int i = 0; i < size_of_grid; i++) {
        for (int j = 0; j < size_of_grid; j++)
            cout << arr[i][j] << " \t";
        cout << "\n";
    }
}
```

SECTION 4:

Time Complexity:

T(n) =4T(n/2) +C

The problem is divided into 4 subproblem (ie) 8x8 problem divides into 4 subproblems of 4x4 which further divides into base case 4 2x2 subproblems there by forming 16 subproblems

By Using Master Theorem, T(n) =a T(n/b) +f(n)

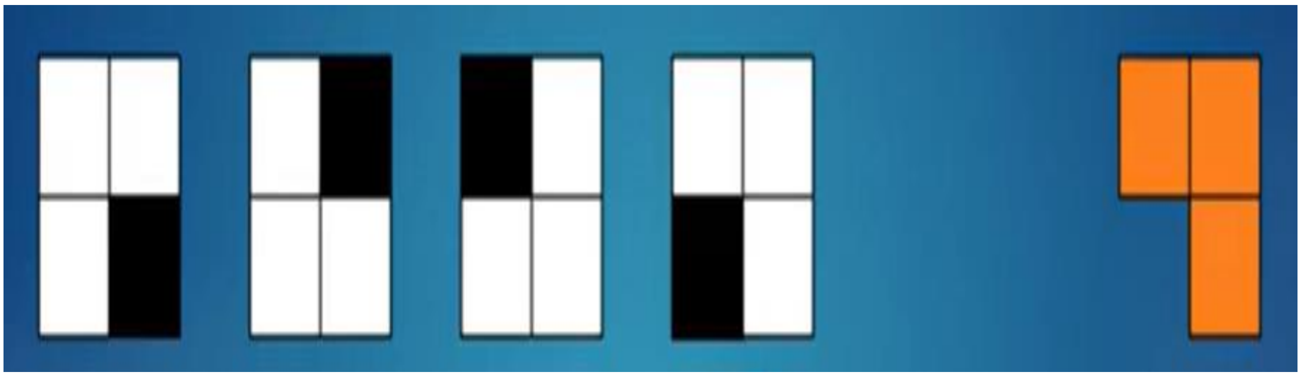
F(n)=Θ(n^0 log^0n)

a=4 b=2

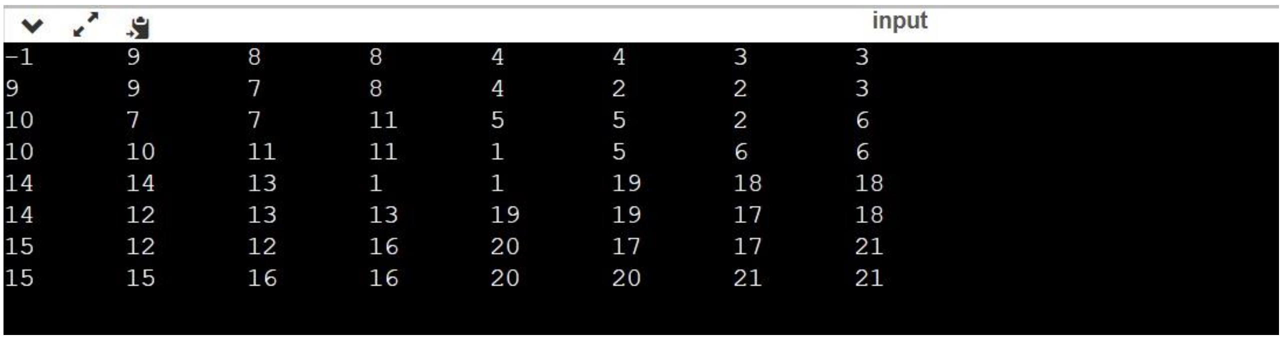
log $_2$ 4 =2>0

There for time complexity =Θ(n^2)

Best case :
If the given chessboard is 2x2 with one tile missing and just 1 single tile covers the chessboard.
Best case time complexity:O(C)



OUTPUT:



```
-1      9       8       8       4       4       3       3
9       9       7       8       4       2       2       3
10      7       7       11      5       5       2       6
10      10      11      11      1       5       6       6
14      14      13      1       1       19      18      18
14      12      13      13      19      19      17      18
15      12      12      16      20      17      17      21
15      15      16      16      20      20      21      21
```

Conclusion:
Defective chessboard concept is implemented successfully and it's complexity is analysed.

References:
https://www.geeksforgeeks.org/tiling-problem-using-divide-and-conquer-algorithm/