

# **Chittagong University of Engineering & Technology**

## **Department of Computer Science & Engineering**

### **Name of the Experiment**

Arranging PCs in L shaped pattern excluding defective PC  
using Divide & Conquer algorithm

**Course No : CSE-244**

**Course Title:** Algorithms Design & Analysis (Sessional)

**Date of experiment :** 27-10-18

**Date of submission :** 04-10-18

Remarks

**Name:** Cinmoy Das

**ID:** 1604092

**Section:** B ; **Group:**B1

**Level:** 2 **Term:** 2

**Objective:**

To arrange some teams in L shape, using divide and conquer algorithm .

**Description:**

There are happening a Programming Contest in a room having sufficient PC. But there is a inactive pc. We have to organize the m number of teams in 'L' shape such that each team stays together.

A defective chessboard is chessboard that has one unavailable (defective) position. Make the other three 4 x 4 chessboards defective by placing a triomino at their common corner.

Recursively tile the four defective 4 x 4 chessboards. The working of recursive divide-and-conquer algorithm can be described by a tree: recursion tree.

The algorithm moves down the recursion tree dividing the large instances into smaller ones. Leaves represent small instances. The recursive algorithm moves back up the tree combining the results from the subtrees. The combining finds the min of the mins computed at leaves and the max of the leaf maxs.

## Source Code:

```
#include<bits/stdc++.h>
using namespace std;
int tile=1;
int arr[100][100];
void chessboard(int fr ,int fc ,int dr ,int dc ,int siz)
{

    if(siz==1)return;
    int tile_no=tile++;
    int s=siz/2;

    if(dr<fr+s&&dc<fc+s)chessboard(fr,fc,dr,dc,s);
    else
    {
        arr[fr+s-1][fc+s-1]=tile_no;
        chessboard(fr,fc,fr+s-1,fc+s-1,s);
    }

    if(dr<fr+s&&dc>=fc+s)chessboard(fr,fc+s,dr,dc,s);
    else
    {
        arr[fr+s-1][fc+s]=tile_no;
        chessboard(fr,fc+s,fr+s-1,fc+s-1,s);
    }

    if(dr>=fr+s&&dc<fc+s)chessboard(fr+s,fc,dr,dc,s);
    else
    {
        arr[fr+s][fc+s-1]=tile_no;
        chessboard(fr+s,fc,fr+s,fc+s-1,s);
    }

    if(dr>=fr+s&&dc>=fc+s)chessboard(fr+s,fc+s,dr,dc,s);
    else
    {
        arr[fr+s][fc+s]=tile_no;
        chessboard(fr+s,fc+s,fr+s,fc+s,s);
    }
}
```

```
}
```

```
int main()
```

```
{
```

```
    int m,k,n,i,j;
```

```
    cin>>m;
```

```
    k=log(3*m)/(2*log(2));
```

```
    n=pow(2,k+1);
```

```
    chessboard(0,0,1,1,n);
```

```
    for(i=0;i<n;i++)
```

```
    {
```

```
        for(j=0;j<n;j++)
```

```
        {
```

```
            cout<<arr[i][j]<<"  ";
```

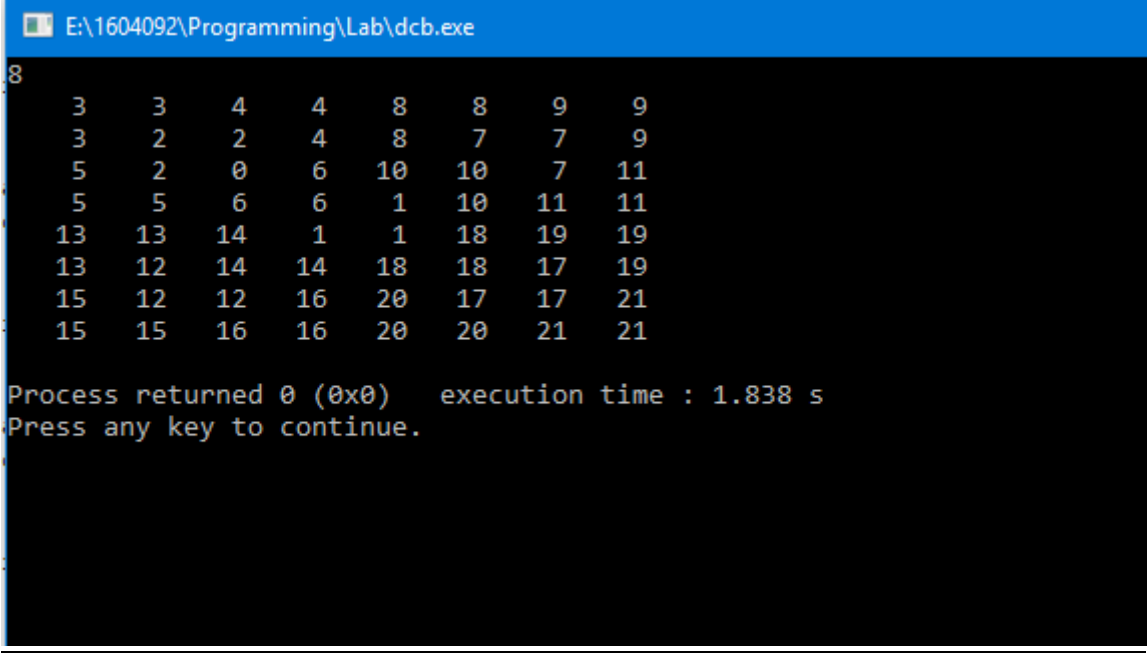
```
        }
```

```
        cout<<endl;
```

```
    }
```

```
}
```

## Input and Output:



```
E:\1604092\Programming\Lab\dcb.exe
8
 3   3   4   4   8   8   9   9
 3   2   2   4   8   7   7   9
 5   2   0   6  10  10   7  11
 5   5   6   6   1  10  11  11
13  13  14   1   1  18  19  19
13  12  14  14  18  18  17  19
15  12  12  16  20  17  17  21
15  15  16  16  20  20  21  21

Process returned 0 (0x0)   execution time : 1.838 s
Press any key to continue.
```

## Analysis of Code :

I have solved the problem using 'Divide and Conquer' algorithm. This algorithm uses recursion. The output was seen to be correct. The time complexity of this code is  $O(n^2)$ .