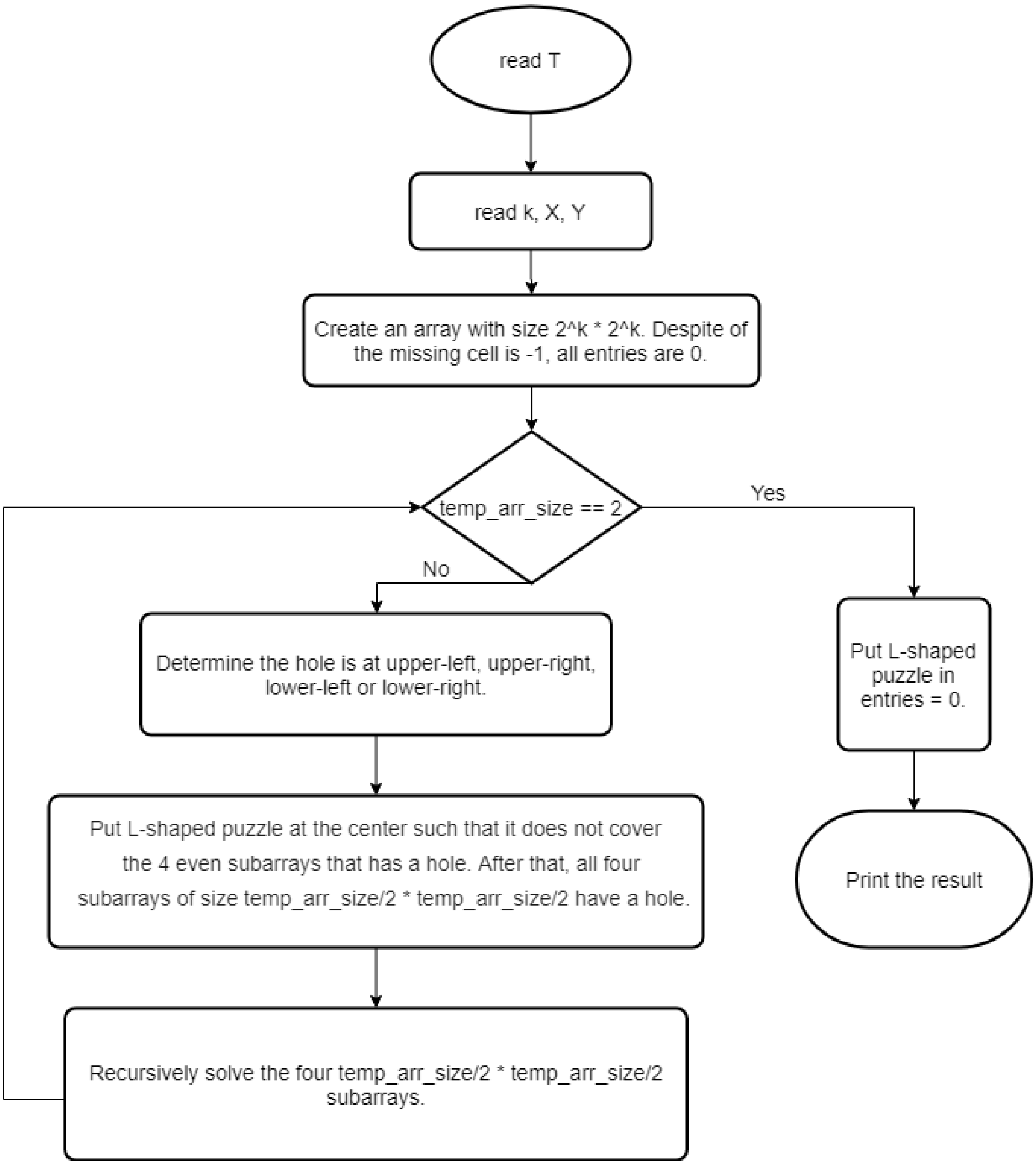
106030012 廖昱瑋 工院21

一、Flowchart


read T
read k, X, Y
Create an array with size 2^k * 2^k. Despite of the missing cell is -1, all entries are 0.
temp_arr_size == 2
Yes
No
Determine the hole is at upper-left, upper-right, lower-left or lower-right.
Put L-shaped puzzle in entries = 0.
Put L-shaped puzzle at the center such that it does not cover the 4 even subarrays that has a hole. After that, all four subarrays of size temp_arr_size/2 * temp_arr_size/2 have a hole.
Print the result
Recursively solve the four temp_arr_size/2 * temp_arr_size/2 subarrays.

二、Time Complexity

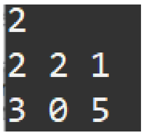
In puzzle initialization and print result part, it takes O(W^2) time.

I use divide and conquer to solve this problem. At each recursive, the program divides the puzzle to 4 pieces of W/2 * W/2 puzzles. In each recursive function takes O(1) time. The total time for recursion part is T(W) = 4T(W / 2) + O(1). By master theorem, this solves to O(W^2).

Therefore, the total time complexity for the whole problem is $O(W^2)$. Since, $W = 2^k$,it is also $O(4^k)$.

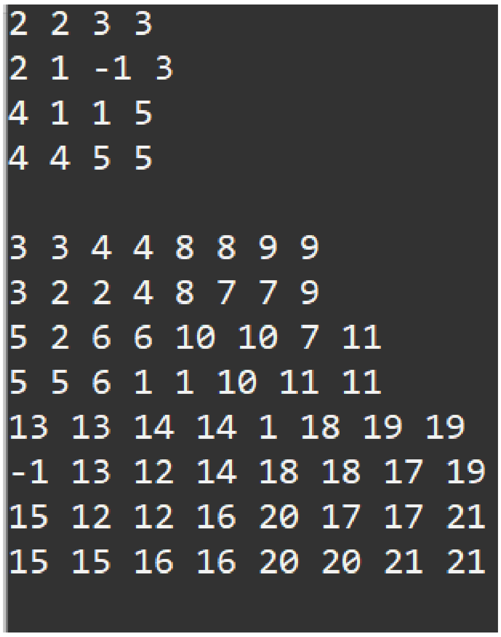
三、Sample Result

Input：



```
2
2 2 1
3 0 5
```

Output：



```
2 2 3 3
2 1 -1 3
4 1 1 5
4 4 5 5

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