#2048(Easy)

이 문제에서 다루는 2048 게임은 보드의 크기가 N×N 이다. 보드의 크기와 보드판의 블록 상태가 주어졌을 때, 최대 5번 이동해서 만들 수 있는 가장 큰 블록의 값을 구하는 프로그램을 작성하시오.

최대 5번 이동시켜서 얻을 수 있는 가장 큰 블록을 출력한다.

이 문제를 풀 때에 당연히 DFS를 이용해야 할 것은 파악이 갔으나 최적의 시간 복잡도를 적용해서 보드의 블록들이 이동할 수 있도록 해야 했다. 그렇게 하기 위해서는 사방의 각 경우별로 나누어서 이동을 시켜야 했었다. 우선 0이 있는 부분에는 0이 없도록 한쪽으로 전부 밀어주었는데, 그렇게 한 뒤에 이동을 시켜야 했으며 앞이나 뒤로 기울이는 경우에는 90도만큼 회전해서 오른쪽, 왼쪽으로 기울이는 방법과 같은 과정을 겪어야 한다.

import sys  
import copy  
n = int(input())  
board = [list(map(int, input().split())) for x in range(n)]  
answer = 0  
dx,dy = [0,0,1,-1], [-1,1,0,0]  
def DFS(a,file, count):  
 global answer  
 new = []  
 if a == 0 or a == 1: #앞, 뒤로 기울이는 경우  
 file = list(map(list, zip(\*file))) #시계방향 90도 회전  
 for i in range(len(file)):  
 row = file[i]  
 notzero = [k for k in row if k != 0]  
 if a == 0 or a == 3: #앞, 왼쪽으로 기울이는 경우  
 for i in range(1, len(notzero)):  
 if notzero[i-1] == notzero[i]:  
 notzero[i-1] += notzero[i]  
 notzero[i] = 0  
 notzero = [k for k in notzero if k != 0]  
 for i in range(len(row) - len(notzero)):  
 notzero.append(0) #부족한 0 채워주기  
 elif a == 1 or a == 2: #앞, 오른쪽으로 기울이는 경우  
 for i in range(len(notzero)-1, 0,-1):  
 if notzero[i-1] == notzero[i]:  
 notzero[i] += notzero[i]  
 notzero[i-1] = 0  
 notzero = [k for k in notzero if k != 0]  
 notzero = notzero[::-1]  
 for i in range(len(row)- len(notzero)):  
 notzero.append(0) #부족한 0을 채워준다  
 notzero = notzero[::-1] #각각의 리스트를 reverse해줌  
 new.append(notzero)  
 if a == 0 or a == 1:  
 new = list(map(list, zip(\*new)))  
 if count == 5:  
 answer = max(answer, max(sum(new, [])))  
 return  
 for k in range(4):  
 DFS(k, new,count+1)  
newboard = copy.deepcopy(board)  
for i in range(4):  
 DFS(i, newboard, 1)  
print(answer)

이러한 방법으로 사방으로 기울이는 것 별로 나누어 코드를 짠 후에 5번 이동 했다면 탈출하도록 한다.