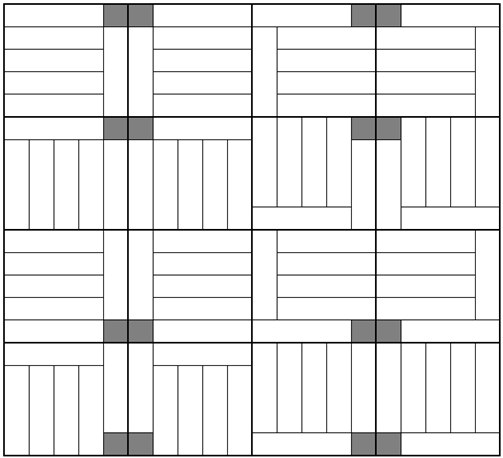
Price Tag  
Cho một hình vuông NxN ô nhỏ (5 ≤ N ≤ 9), mỗi ô chứa 1 số từ 0 – 9.  
Nhiệm vụ cần làm là cắt hình vuông lớn thành (N+1) hình có kích thước 1x(N-1). Mỗi hình kích thước 1x(N-1) bao gồm (N-1) chữ số và tạo thành 1 số nguyên (N-1) chữ số (đọc từ trái qua phải hoặc từ trên xuống dưới tùy theo phương án cắt).   
  
Sau khi có được các miếng cắt, tính tổng các số thu được (chỉ tính các số có N-1 chữ số, ô còn sót lại sẽ không được tính vào tổng).   
Hãy tìm ra phương án cắt có tổng thu được lớn nhất.  
  
  
  
Input  
Input sẽ chứa một hoặc nhiều test cases   
Dòng đầu là số lượng test case T (T ≤ 50)  
Dòng đầu tiên của mỗi test cases là kích thước hình vuông N (5 ≤ N ≤ 9)  
N dòng tiếp theo lần lượt là giá trị trong các ô nhỏ của hình vuông lớn.   
  
Output  
Với mỗi test case, in ra tổng số lớn nhất thu được   
  
Sample  
Input  
2  
5  
1 0 3 5 6  
4 5 3 7 9  
7 2 2 6 7  
3 4 7 8 8  
9 5 1 6 3  
6  
3 4 5 6 8 6  
3 5 6 7 7 8  
3 3 3 3 3 5  
1 2 5 6 4 2  
3 3 5 5 5 7  
8 5 3 6 9 9  
  
Output  
#1 35000  
#2 72000

input Price Tag

50  
5  
0 3 7 0 3   
3 4 7 1 7   
7 3 3 0 6   
5 1 0 5 5   
2 6 0 7 2   
5  
1 1 5 8 5   
4 2 1 5 0   
3 5 2 3 3   
0 8 0 4 3   
2 7 2 3 1   
5  
1 3 1 8 6   
3 4 7 6 6   
3 5 5 3 5   
0 1 3 1 3   
5 3 0 1 8   
5  
5 4 5 8 6   
6 8 8 7 7   
6 4 2 0 8   
4 8 0 6 0   
2 4 8 7 6   
5  
1 7 1 1 4   
1 7 4 0 2   
6 7 4 2 2   
4 5 6 8 4   
5 2 1 3 1   
5  
3 2 4 3 3   
4 5 0 3 5   
2 6 4 4 6   
1 0 0 7 1   
4 3 3 1 3   
5  
5 2 7 7 8   
1 8 8 8 8   
0 1 3 1 4   
2 5 3 0 0   
7 1 0 8 8   
5  
1 0 2 5 1   
8 4 5 1 4   
6 5 1 0 0   
7 4 5 7 1   
1 1 4 0 2   
6  
6 0 6 3 1 5   
0 3 1 6 4 0   
2 0 6 3 4 4   
8 6 5 3 7 1   
3 4 3 7 7 8   
0 1 6 0 1 3   
6  
3 7 6 1 0 8   
2 0 5 8 3 1   
1 0 4 4 0 7   
5 5 4 4 6 5   
2 7 6 7 2 7   
4 6 4 1 6 8   
6  
3 8 7 0 8 0   
7 3 0 8 4 3   
5 1 2 0 3 1   
8 4 5 4 7 2   
0 0 4 3 8 7   
7 7 7 0 0 5   
6  
2 1 4 1 3 3   
5 6 1 8 5 0   
1 6 5 0 1 6   
5 4 0 8 4 2   
0 0 3 3 3 4   
3 1 1 0 4 6   
6  
1 3 2 5 4 8   
4 2 8 6 3 1   
3 7 5 5 4 4   
6 1 2 6 1 7   
4 6 6 6 1 1   
7 5 3 5 5 8   
6  
2 8 6 1 1 4   
2 6 7 8 3 7   
6 1 2 0 8 6   
6 0 6 7 5 4   
7 7 2 6 6 1   
7 6 6 3 4 6   
7  
7 0 0 3 4 7 0   
4 3 1 1 4 6 4   
1 1 3 5 6 5 1   
1 7 3 2 1 0 0   
4 6 6 2 0 6 2   
7 1 5 3 5 2 8   
4 8 8 2 8 0 0   
7  
4 7 3 4 4 4 5   
5 7 1 5 1 6 8   
3 0 2 5 6 3 1   
8 8 3 4 2 5 6   
8 5 4 2 8 1 1   
6 3 5 8 0 4 0   
7 8 6 2 0 7 8   
7  
1 0 6 5 6 6 0   
2 7 0 7 4 7 3   
1 3 1 7 3 5 3   
6 2 6 3 2 2 3   
4 8 1 6 3 8 0   
8 0 7 2 2 3 0   
4 0 1 5 6 6 8   
7  
5 7 4 5 7 7 1   
5 2 0 2 2 1 4   
2 0 7 8 3 5 1   
6 3 0 8 2 1 4   
3 4 6 0 2 0 8   
6 5 3 4 2 6 3   
3 2 5 1 6 1 4   
7  
3 2 3 5 8 7 0   
5 8 6 4 0 7 0   
3 2 4 4 4 8 8   
2 4 5 1 3 3 6   
8 3 7 3 2 8 5   
1 6 4 5 4 4 8   
3 2 1 4 7 8 7   
7  
3 1 0 5 2 7 2   
3 0 5 4 1 0 5   
7 8 8 2 1 4 8   
4 4 2 8 7 3 6   
5 1 7 3 8 0 7   
1 6 1 8 7 8 3   
2 0 1 6 6 0 1   
7  
0 3 3 0 0 1 6   
4 2 8 6 2 5 7   
0 6 1 4 5 0 3   
1 8 8 6 2 8 3   
5 1 5 5 7 4 8   
6 5 7 2 8 2 0   
8 2 8 7 3 4 8   
7  
4 4 2 3 7 0 8   
5 6 2 0 0 3 4   
4 4 3 6 3 1 1   
7 5 1 6 3 6 1   
0 0 3 0 2 1 1   
0 5 3 1 6 0 1   
7 7 5 0 1 8 1   
7  
5 4 2 1 2 7 8   
7 2 4 2 3 4 8   
6 1 5 3 4 5 4   
3 4 6 0 5 6 3   
6 4 3 0 0 8 6   
1 1 4 2 3 0 6   
7 1 8 5 7 1 2   
7  
1 8 5 8 6 0 7   
3 5 0 8 5 2 1   
4 0 2 8 1 0 4   
8 2 0 0 7 4 8   
7 4 8 3 4 2 4   
6 3 4 1 7 7 3   
6 1 5 6 8 8 0   
7  
0 8 0 8 1 6 4   
8 7 6 2 7 3 7   
7 7 5 8 0 7 8   
6 2 1 8 1 6 0   
6 2 4 8 8 4 8   
6 5 8 6 8 6 5   
5 8 0 4 7 0 2   
7  
0 3 8 7 8 7 3   
8 2 1 7 4 2 6   
3 8 6 7 5 3 3   
3 6 8 8 1 7 7   
3 8 1 6 0 3 8   
8 8 1 8 0 8 1   
4 2 4 3 5 3 7   
7  
8 6 3 2 4 3 6   
0 7 4 3 5 3 2   
2 3 5 3 3 4 3   
6 2 8 8 4 1 4   
7 8 4 8 4 8 5   
6 0 2 8 1 6 4   
1 3 2 7 0 7 7   
8  
6 4 7 7 6 0 0 4   
5 1 2 0 3 5 6 4   
8 1 2 2 8 0 0 8   
4 1 2 4 7 2 2 4   
7 3 0 2 8 7 6 3   
6 4 5 6 0 7 6 3   
2 3 5 8 7 8 4 8   
1 1 4 1 1 5 7 2   
8  
5 2 7 4 6 5 8 1   
1 6 1 7 1 6 7 2   
0 7 3 6 6 0 2 2   
6 8 3 7 3 8 2 6   
8 6 5 7 6 5 1 4   
1 3 8 5 0 0 2 6   
6 7 4 5 4 2 8 4   
4 8 6 2 6 4 5 6   
8  
2 3 0 2 2 1 7 3   
7 8 6 0 6 0 1 7   
3 3 6 0 1 2 1 3   
5 1 5 8 3 8 5 2   
2 4 2 8 6 2 8 7   
8 7 7 5 7 5 1 1   
6 3 4 4 1 2 8 6   
7 5 5 0 6 0 5 4   
8  
3 6 6 7 5 1 8 7   
2 3 7 1 6 6 0 5   
8 4 5 5 3 2 5 4   
3 2 8 4 5 1 1 0   
3 8 6 6 0 5 1 4   
4 4 1 8 1 3 7 6   
8 6 8 7 8 8 0 2   
4 2 0 7 6 0 5 3   
8  
5 2 8 2 0 3 8 8   
0 3 4 7 8 7 4 7   
7 1 2 8 1 6 4 0   
8 2 7 4 3 8 1 1   
1 8 3 4 6 7 4 3   
5 6 4 4 7 2 6 5   
6 1 6 7 2 4 5 6   
3 0 4 0 5 7 0 7   
8  
3 2 8 1 0 3 8 5   
7 3 8 7 3 3 7 8   
8 1 7 1 8 4 6 8   
8 6 8 0 0 2 4 4   
4 2 2 7 7 5 3 7   
1 6 4 4 8 5 4 7   
4 3 5 5 7 3 0 4   
6 8 2 3 7 2 8 0   
8  
1 1 0 8 4 6 2 6   
0 3 2 3 2 4 3 0   
0 3 7 5 8 4 1 8   
2 3 2 3 6 3 0 7   
5 3 1 8 7 0 7 5   
7 1 1 8 4 5 3 1   
0 2 4 2 4 5 1 2   
8 8 4 3 6 3 1 6   
8  
8 2 0 4 8 5 7 6   
4 8 7 5 3 8 0 8   
2 5 4 8 4 1 2 1   
7 1 0 8 5 6 2 3   
4 4 1 8 3 6 0 0   
3 3 4 1 4 4 2 6   
8 1 6 1 0 7 3 1   
8 4 0 1 7 0 3 2   
8  
0 8 4 1 2 5 2 8   
8 4 6 8 6 3 7 0   
3 4 8 2 2 3 8 6   
0 5 7 8 3 4 1 8   
2 7 2 8 8 1 7 5   
3 2 4 4 5 1 3 1   
7 5 3 7 4 8 8 5   
5 2 6 0 5 6 6 8   
8  
0 4 1 6 3 5 4 1   
7 0 0 3 0 2 4 4   
7 8 0 3 7 7 4 0   
1 8 0 0 0 1 1 8   
7 3 3 3 1 6 0 5   
1 4 8 8 6 0 7 0   
8 8 3 2 8 2 2 8   
5 2 0 6 2 6 8 4   
8  
6 8 2 2 7 7 4 7   
4 6 7 3 7 8 7 0   
8 0 6 0 6 2 1 2   
0 0 6 0 6 7 0 6   
1 7 1 3 3 4 3 0   
5 0 2 5 7 1 0 2   
6 8 7 5 1 2 3 2   
7 2 3 0 6 2 1 5   
8  
0 6 6 5 0 1 4 0   
5 3 1 6 1 6 5 5   
7 6 7 2 6 4 7 7   
0 3 4 5 4 0 8 1   
6 4 1 6 4 8 7 4   
4 4 5 4 4 8 7 0   
8 3 7 3 0 3 1 1   
2 1 6 4 4 6 0 0   
9  
2 0 2 4 2 1 4 1 7   
2 6 0 2 6 5 2 6 5   
1 5 6 5 7 4 2 6 6   
6 1 4 7 4 8 8 4 0   
2 7 6 1 5 8 0 0 5   
6 1 7 2 0 0 0

3 4   
5 5 4 8 7 3 4 6 6   
4 0 3 8 2 6 3 8 2   
1 4 4 2 0 3 4 4 3   
9  
4 8 1 3 2 0 7 8 5   
3 0 5 5 7 6 1 3 2   
5 7 6 2 5 6 5 2 7   
6 6 5 2 3 1 2 4 4   
5 7 5 6 2 0 0 4 2   
8 0 4 1 0 8 3 3 0   
6 0 5 6 0 8 6 7 1   
2 1 5 0 5 3 1 1 6   
5 7 2 5 8 3 4 4 6   
9  
4 3 0 7 2 2 2 6 0   
8 1 0 5 1 4 6 2 2   
3 5 7 8 4 2 1 2 2   
3 8 8 6 7 1 0 8 7   
4 3 3 3 1 7 0 3 5   
3 7 4 1 7 7 3 5 4   
3 8 2 2 5 1 3 3 7   
0 3 1 1 4 5 1 0 3   
2 4 0 5 6 2 1 8 2   
9  
5 2 2 5 5 6 3 2 4   
8 7 8 8 8 0 3 3 2   
4 4 7 4 3 7 6 5 2   
6 8 1 2 4 7 8 4 8   
0 4 4 1 7 0 0 7 3   
3 5 2 6 4 3 3 3 0   
4 6 8 7 4 4 4 8 7   
1 3 3 1 3 5 0 4 0   
3 8 1 6 7 8 8 8 5   
9  
3 1 1 6 6 7 6 1 8   
7 2 8 5 6 2 2 1 3   
3 5 2 3 8 6 0 0 8   
2 0 2 3 2 1 8 8 4   
1 1 4 5 4 5 4 3 1   
5 8 6 0 3 4 3 3 4   
3 7 3 6 3 3 0 2 4   
6 2 8 5 6 3 3 8 5   
7 2 1 7 5 8 2 5 3   
9  
4 4 1 0 6 3 7 1 3   
5 2 0 6 3 6 1 5 8   
1 7 0 4 5 5 0 4 6   
0 3 1 0 3 6 4 0 6   
3 0 5 6 6 6 6 1 3   
8 3 1 7 0 3 0 3 5   
5 3 3 5 3 3 4 7 8   
1 3 3 7 2 3 5 1 6   
4 5 8 6 0 8 6 6 5   
9  
5 3 6 4 5 6 5 6 7   
6 3 7 0 6 0 5 5 8   
4 1 5 5 1 4 8 8 6   
6 1 0 1 7 2 3 4 4   
6 4 0 7 6 0 0 2 8   
2 6 8 3 0 1 5 8 7   
0 7 2 6 7 0 8 2 4   
2 1 6 5 5 1 5 0 0   
0 7 8 5 6 4 5 2 1   
9  
6 8 6 4 8 0 5 2 3   
7 2 3 8 0 3 7 2 2   
5 4 1 5 2 3 1 2 3   
1 6 1 8 3 1 5 1 4   
3 0 8 6 0 2 3 6 0   
6 1 2 4 4 5 4 2 4   
6 8 5 1 6 5 4 2 2   
5 3 0 0 3 2 1 8 3   
5 5 5 1 4 1 6 0 2   
9  
8 0 0 3 0 6 4 1 6   
2 4 4 5 6 5 0 1 2   
5 5 1 8 4 5 5 4 6   
3 5 7 2 8 5 3 4 1   
2 6 7 5 5 6 1 8 6   
0 7 8 6 3 4 4 2 3   
6 3 4 7 5 2 5 7 1   
0 5 0 0 4 7 3 1 6   
8 0 1 4 3 5 3 8 7   
9  
6 4 1 0 3 2 2 3 4   
4 3 6 5 7 6 8 5 1   
2 7 6 3 0 3 2 4 4   
2 6 5 6 4 7 4 3 0   
4 7 6 6 4 1 8 5 5   
2 2 8 3 8 5 6 4 0   
1 3 5 1 2 3 0 2 5   
6 2 4 8 8 0 5 1 0   
0 7 5 3 7 0 6 6 0   
9  
6 2 7 7 2 7 2 0 3   
2 6 3 6 1 8 3 1 5   
2 2 3 2 2 5 0 5 5   
8 1 8 8 3 8 1 6 1   
7 2 5 5 2 6 6 6 6   
1 6 3 1 0 5 5 4 6   
3 7 7 5 3 2 2 7 1   
0 3 8 7 8 2 4 4 7   
0 7 3 2 3 4 4 4 6

output Price Tag

#1 29868  
#2 31478  
#3 31383  
#4 43665  
#5 33999  
#6 23312  
#7 40877  
#8 28595  
#9 271774  
#10 335089  
#11 405327  
#12 317236  
#13 358419  
#14 465096  
#15 3989426  
#16 5584349  
#17 3426996  
#18 4387171  
#19 3681539  
#20 3314673  
#21 3959856  
#22 3874076  
#23 4587937  
#24 4578980  
#25 5726952  
#26 4932907  
#27 4139966  
#28 45947753  
#29 56272197  
#30 51827556  
#31 50595292  
#32 50397277  
#33 53942284  
#34 38933551  
#35 55190401  
#36 54415336  
#37 47716291  
#38 58363137  
#39 47106835  
#40 428415669  
#41 548202944  
#42 544896370  
#43 624796692  
#44 502007781  
#45 438800829  
#46 601277724  
#47 521739288  
#48 547668409  
#49 576658347  
#50 473877240

--- Wednesday, July 31, 2019 ---

Giải thích:   
Như ở hình vuông đầu tiên, ô đen ở góc cùng trên phải.  
Các phần sẽ lấy là  
1035 + 4537 + 7226 + 3478 + 9516 + 9783;  
Làm tương tự với các hình còn lại

[#include](tg://search_hashtag?hashtag=include) <iostream>  
  
using namespace std;  
int Arr[10][10];  
int Visited[10][10];  
int Number[10][9];  
int N;  
int index;  
bool flag;  
int Answer;  
int fx[9] = {0,2,3,4,5,0,0,0,0};  
  
int chuyendoi(int day[9], int n){  
int a = 0;  
for(int i = 0; i < n; i++){  
a = a\*10+day[i];  
}  
return a;  
}  
void backtrack(int step, int cnt){  
int x,y;  
if(step == N+1){  
int sum = 0;  
for(int i = 0; i <= N; i++){  
sum = sum + chuyendoi(Number[i], N-1);  
}  
if(sum > Answer)  
Answer = sum;  
return;  
}  
if(cnt > (N\*(N-1)+1))  
return;  
else{  
bool flag = true;  
x = cnt/N;  
y = cnt%N;  
  
for(int t = 0; t < 3; t++){  
//doc  
if(t == 0 && x < 2){  
flag = true;  
for(int i = x; i < x + N-1; i++){  
if(Visited[i][y] == 1){  
flag = false;  
break;  
}  
}  
if(flag){  
index = 0;  
for(int i = x; i < x + N-1; i++){  
Visited[i][y] = 1;  
Number[step][index] = Arr[i][y];  
index++;  
}  
backtrack(step+1, cnt+1);  
for(int i = x; i < x + N-1; i++){  
Visited[i][y] = 0;  
}  
}  
}  
//ngang  
else if(t == 1 && y < 2){  
flag = true;  
for(int i = y; i < y + N-1; i++){  
if(Visited[x][i] == 1){  
flag = false;  
break;  
}  
}  
if(flag){  
index = 0;  
for(int i = y; i < y + N-1; i++){  
Visited[x][i] = 1;  
Number[step][index] = Arr[x][i];  
index++;  
}  
backtrack(step+1, cnt+N-1);  
for(int i = y; i < y + N-1; i++){  
Visited[x][i] = 0;  
}  
}  
}  
else if(t == 2){  
backtrack(step, cnt +1);  
}  
}  
  
  
}  
  
}  
int main()  
{  
int tc,TestCase, output=0;  
  
freopen("Text.txt", "r", stdin);  
freopen("Text1.txt", "w", stdout);  
  
cin>>TestCase;  
  
for(tc=0; tc<TestCase; tc++)  
{  
output = 0;  
cin >> N;  
for(int i = 0; i < N; i++){  
for(int j = 0; j < N; j++){  
cin >> Arr[i][j];  
}  
}  
  
  
Answer = 0;  
backtrack(0,0);  
////////////////////////////////  
  
  
  
////////////////////////////////  
  
  
  
cout<< "#" << tc+1<<" "<< Answer <<endl;  
}  
  
return 0;  
}

package PriceTag;  
  
import java.io.FileInputStream;  
import java.util.Scanner;  
  
public class newpricetag {  
static int N;  
static int[][] maze;  
static int[] Dx = { 1, 0, 0 };  
static int[] Dy = { 0, 1, 0 };  
static int[][] Vs;  
static int max;  
static String temp;  
static boolean had;  
  
public static void main(String args[]) throws Exception {  
System.setIn(new FileInputStream("src/PriceTag/input"));  
Scanner sc = new Scanner([System.in](http://system.in/));  
int T = sc.nextInt();  
for (int test = 1; test <= T; test++) {  
N = sc.nextInt();  
maze = new int[N][N];  
Vs = new int[N][N];  
for (int i = 0; i < N; i++) {  
for (int j = 0; j < N; j++) {  
maze[i][j] = sc.nextInt();  
}  
}  
max = 0;  
BT(0, 0, 0);  
System.out.println("#" + test + " " + max);  
}  
}  
  
static void BT(int x, int y, int total) {  
if (x == N - 1 && y == N - 1) {  
if (total > max) {  
max = total;  
}  
return;  
}  
if (Vs[x][y] == 1) {  
if (y == N - 1)  
BT(x + 1, 0, total);  
else {  
BT(x, y + 1, total);  
}  
} else {  
for (int k = 0; k < 3; k++) {  
if (k != 2) {  
int a = x + Dx[k] \* (N - 2);  
int b = y + Dy[k] \* (N - 2);  
if (a <= N - 1 && b <= N - 1) {  
had = true;  
for (int i = 1; i < N - 1; i++) {  
if (Vs[x + Dx[k] \* i][y + Dy[k] \* i] == 1) {  
had = false;  
break;  
}  
}  
if (had) {  
temp = "0";  
for (int i = 0; i < N - 1; i++) {  
Vs[x + Dx[k] \* i][y + Dy[k] \* i] = 1;  
temp = temp + String.valueOf(maze[x + Dx[k] \* i][y + Dy[k] \* i]);  
}  
if (y == N - 1) {  
BT(x + 1, 0, total + Integer.parseInt(temp));  
} else  
BT(x, y + 1, total + Integer.parseInt(temp));  
for (int i = 0; i < N - 1; i++) {  
Vs[x + Dx[k] \* i][y + Dy[k] \* i] = 0;  
}  
} else {  
if (y == N - 1) {  
BT(x + 1, 0, total);  
} else  
BT(x, y + 1, total);  
}  
}  
}   
else if ((x == 0 && y == N - 1) || (x == 0 && y == 0) || (x == N - 1 && y == N - 1)  
|| (x == N - 1 && y == 0)) {  
if (y == N - 1) {  
BT(x + 1, 0, total);  
} else  
BT(x, y + 1, total);  
}  
}  
}  
}  
}

package PriceTag;  
  
import java.io.FileInputStream;  
import java.util.Scanner;  
  
public class newpricetag {  
static int N;  
static int[][] maze;  
static int[] Dx = { 1, 0, 0 };  
static int[] Dy = { 0, 1, 0 };  
static int[][] Vs;  
static int max;  
static String temp;  
static boolean had;  
  
public static void main(String args[]) throws Exception {  
System.setIn(new FileInputStream("src/PriceTag/input"));  
Scanner sc = new Scanner([System.in](http://system.in/));  
int T = sc.nextInt();  
for (int test = 1; test <= T; test++) {  
N = sc.nextInt();  
maze = new int[N][N];  
Vs = new int[N][N];  
for (int i = 0; i < N; i++) {  
for (int j = 0; j < N; j++) {  
maze[i][j] = sc.nextInt();  
}  
}  
max = 0;  
BT(0, 0, 0);  
System.out.println("#" + test + " " + max);  
}  
}  
  
static void BT(int x, int y, int total) {  
if (x == N - 1 && y == N - 1) {  
if (total > max) {  
max = total;  
}  
return;  
}  
if (Vs[x][y] == 1) {  
if (y == N - 1)  
BT(x + 1, 0, total);  
else {  
BT(x, y + 1, total);  
}  
} else {  
for (int k = 0; k < 3; k++) {  
if (k != 2) {  
int a = x + Dx[k] \* (N - 2);  
int b = y + Dy[k] \* (N - 2);  
if (a <= N - 1 && b <= N - 1) {  
had = true;  
for (int i = 1; i < N - 1; i++) {  
if (Vs[x + Dx[k] \* i][y + Dy[k] \* i] == 1) {  
had = false;  
break;  
}  
}  
if (had) {  
temp = "0";  
for (int i = 0; i < N - 1; i++) {  
Vs[x + Dx[k] \* i][y + Dy[k] \* i] = 1;  
temp = temp + String.valueOf(maze[x + Dx[k] \* i][y + Dy[k] \* i]);  
}  
if (y == N - 1) {  
BT(x + 1, 0, total + Integer.parseInt(temp));  
} else  
BT(x, y + 1, total + Integer.parseInt(temp));  
for (int i = 0; i < N - 1; i++) {  
Vs[x + Dx[k] \* i][y + Dy[k] \* i] = 0;  
}  
} else {  
if (y == N - 1) {  
BT(x + 1, 0, total);  
} else  
BT(x, y + 1, total);  
}  
}  
}   
else if ((x == 0 && y == N - 1) || (x == 0 && y == 0) || (x == N - 1 && y == N - 1)  
|| (x == N - 1 && y == 0)) {  
if (y == N - 1) {  
BT(x + 1, 0, total);  
} else  
BT(x, y + 1, total);  
}  
}  
}  
}  
}

package luyende;  
  
import java.io.FileInputStream;  
import java.util.Scanner;  
  
/\*   
\* Tóm lại bài này sai ở: điều kiện xét visit vs điều kiện khi backTrack nhảy ở cột (n-1) xuống hàng tiếp theo. 2 lỗi nhỏ này thôi cũng làm hỏng cả chương trình của mình đó.  
\* bài này giống bài Cover rectangle with dominos  
\* Thế éo nào bài này mình bị bug nhiều thế nhỉ  
\* copy lại rồi chỉnh sửa chứng tỏ của mình ko phải bị sai ở chỗ trong chỗ xử lý backTrack mà là ở chỗ khác  
\* sửa chỗ điều kiện dừng vs đã visit mà max lên được, chứng tỏ mình code sai gì đó ở đoạn này (tới đoạn a<=n-1 && b<=n-1  
\*/  
public class PriceTag {  
Scanner sc = new Scanner([System.in](http://system.in/));  
int t, N, max;  
int maze[][], Vs[][];  
int Dx[] = { 0, 1, 0 };  
int Dy[] = { 1, 0, 0 };  
  
// (0,0) đứng yên vị trí  
void init() {  
max = 0;  
maze = new int[N][N];  
Vs = new int[N][N];  
}  
  
void BT(int x, int y, int total) {  
if (x == N - 1 && y == N - 1) {  
if (total > max) {  
max = total;  
}  
return;  
}  
if (Vs[x][y] == 1) {  
if (y == N - 1) {  
BT(x + 1, 0, total);  
} else {  
BT(x, y + 1, total);  
}  
}   
// else chỗ này nghĩa là chưa visit, mọi bài mình ko có else chả qua là xét visit ở trong for, còn bài này mình xét visit là else trước for  
else { // nếu ko để else ở đây thì chương trình sẽ ko thể hiểu được là điểm đang xét chưa visit  
for (int k = 0; k < 3; k++) {  
if (k != 2) {  
int a = x + Dx[k] \* (N - 2);  
int b = y + Dy[k] \* (N - 2);  
if (a <= N - 1 && b <= N - 1) {  
boolean had = true;  
for (int i = 1; i < N - 1; i++) {  
if (Vs[x + Dx[k] \* i][y + Dy[k] \* i] == 1) {  
had = false;  
break;  
}  
}  
if (had) {  
String temp = "0";  
for (int i = 0; i < N - 1; i++) {  
Vs[x + Dx[k] \* i][y + Dy[k] \* i] = 1;  
temp = temp + String.valueOf(maze[x + Dx[k] \* i][y + Dy[k] \* i]);  
}  
if (y == N - 1) {  
BT(x + 1, 0, total + Integer.parseInt(temp));  
} else  
BT(x, y + 1, total + Integer.parseInt(temp));  
for (int i = 0; i < N - 1; i++) {  
Vs[x + Dx[k] \* i][y + Dy[k] \* i] = 0;  
}  
} else {  
if (y == N - 1) {  
BT(x + 1, 0, total);  
} else {  
BT(x, y + 1, total);  
}  
}  
}  
}  
// 4 góc ma trận  
else {  
if ((x == 0 && y == 0) || (x == 0 && y == N - 1) || (x == N - 1 && y == 0)  
|| (x == N - 1 && y == N - 1)) {  
if (y == N - 1) {  
BT(x + 1, 0, total);  
} else {  
BT(x, y + 1, total);  
}  
}  
}  
}  
}  
  
}  
  
void solution() {  
t = sc.nextInt();  
for (int tc = 1; tc <= t; tc++) {  
N = sc.nextInt();  
init();  
for (int i = 0; i < N; i++) {  
for (int j = 0; j < N; j++) {  
maze[i][j] = sc.nextInt();  
}  
}  
BT(0, 0, 0);  
System.out.println("#" + tc + " " + max);  
}  
}  
  
public static void main(String args[]) throws Exception {  
System.setIn(new FileInputStream("PriceTag.txt"));  
PriceTag priceTag = new PriceTag();  
priceTag.solution();  
}  
}

[5:35:27 PM]Huu Uoc:

// 4 goc cua map. ko chan can thi bi time out cho nay  
else {  
if((x==0 && y==0) || (x==0 && y==n-1) && (x==n-1 && y==0) && (x==n-1 && y==n-1))  
if (y == n - 1) {  
backTrack(x + 1, 0, count);  
} else {  
backTrack(x, y + 1, count);  
}  
}

Top of Form

Bottom of Form