package luyende;  
  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
public class CleaningRobot {  
Scanner sc = new Scanner([System.in](http://system.in/));  
int t, n, m, st, en, L, answer;  
int map[][];  
int visit[][];  
int queueX[] = new int[1000000];  
int queueY[]= new int[1000000];  
int vertex[][];  
int adj[][];  
int visitHoanVi[];  
int dx[] = { 1, -1, 0, 0 };  
int dy[] = { 0, 0, 1, -1 };  
  
void init() {  
visitHoanVi = new int[1000];  
answer = 100000;  
adj = new int[1000][1000];  
vertex = new int[1000][2];  
L = 1;  
  
map = new int[n + 1][m + 1];  
visit = new int[n + 1][m + 1];  
}  
  
void initVisit() {  
for (int i = 0; i < n; i++) {  
for (int j = 0; j < m; j++) {  
visit[i][j] = -1;  
}  
}  
}  
void resultVisit(){  
System.out.println("visit");  
for (int i = 0; i < n; i++) {  
for (int j = 0; j < m; j++) {  
System.out.print(visit[i][j]+" ");  
}  
System.out.println();  
}  
}  
void BFS(int startX, int startY) {  
st = en = 0;  
queueX[en] = startX;  
queueY[en] = startY;  
en++;  
visit[startX][startY] = 0;  
while (st != en) {  
int x = queueX[st];  
int y = queueY[st];  
st++;  
for (int i = 0; i < 4; i++) {  
int xNext = x + dx[i];  
int yNext = y + dy[i];  
if (xNext >= 0 && xNext < n && yNext >= 0 && yNext < m) {  
if (visit[xNext][yNext] == -1) {  
if (map[xNext][yNext] != 2) {  
queueX[en] = xNext;  
queueY[en] = yNext;  
en++;  
visit[xNext][yNext] = visit[x][y] + 1;  
}  
}  
}  
}  
}  
}  
void solve(int idx, int last, int sum) {  
  
if (idx == L) {  
if(sum < answer){  
answer = sum;  
}  
return;  
}  
// vừa xong bị time out chỗ này: để i chạy từ 0 tới L-1 (L=4). Vì trường hợp chạy từ 0 sẽ lắp từ 0 tới 0 tốn thêm thời gian  
for (int i = 1; i < L; i++) {  
if (visitHoanVi[i] == 0 && adj[last][i] != -1) {  
visitHoanVi[i] = 1;  
solve(idx + 1, i, sum + adj[last][i]);  
visitHoanVi[i] = 0;  
}  
}  
}  
  
void solution() {  
t = sc.nextInt();  
for (int tc = 1; tc <= t; tc++) {  
n = sc.nextInt();  
m = sc.nextInt();  
init();  
initVisit();  
for (int i = 0; i < n; i++) {  
for (int j = 0; j < m; j++) {  
map[i][j] = sc.nextInt();  
}  
}  
for (int i = 0; i < n; i++) {  
for (int j = 0; j < m; j++) {  
if (map[i][j] == 3) {  
vertex[0][0] = i;  
vertex[0][1] = j;  
} else if (map[i][j] == 1) {  
vertex[L][0] = i;  
vertex[L][1] = j;  
L++;  
}  
}  
}  
System.out.println("map[][]");  
for (int i = 0; i < n; i++) {  
for (int j = 0; j < m; j++) {  
System.out.print(map[i][j] + " ");  
}  
System.out.println();  
}  
System.out.println("vertex[][]");  
for (int i = 0; i < L; i++) {  
for (int j = 0; j < 2; j++) {  
System.out.print(vertex[i][j] + " ");  
}  
System.out.println();  
}  
for (int i = 0; i < L; i++) {  
initVisit();  
BFS(vertex[i][0], vertex[i][1]);  
resultVisit();  
for (int j = 0; j < L; j++) {  
adj[i][j] = visit[vertex[j][0]][vertex[j][1]];  
}  
}  
  
System.out.println("adj[][]");  
for (int i = 0; i < L; i++) {  
for (int j = 0; j < L; j++) {  
System.out.print(adj[i][j] + " ");  
}  
System.out.println();  
}  
// vẽ ma trận adj[][] ra là thấy  
// bắt đầu từ đỉnh 0 đi tới đỉnh 1 vs tổng ban đầu là 0  
// solve(1, 0, 0);  
// if (answer == 100000) {  
// answer = -1;  
// }  
answer = 1000000;  
solve(1, 0, 0);  
if (answer == 1000000)  
answer = -1;  
System.out.println("Case #" + tc);  
System.out.println(answer);  
}  
}  
  
public static void main(String[] args) throws Exception {  
System.setIn(new FileInputStream("CleaningRobot.txt"));  
CleaningRobot c = new CleaningRobot();  
c.solution();  
}  
  
}

// chú ý kiểm soát biến L