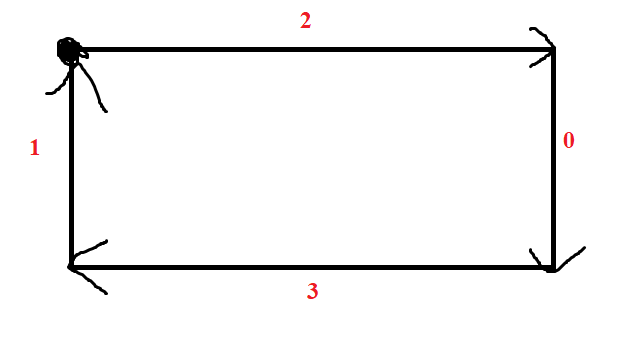
Jogging

Cho một khu vực với kích thước NxN (2<=N<=10) bên trong chứa các ô đất với chiều cao khác nhau. Jack muốn chạy bộ giữa các ô đất này theo các lộ trình hình chữ nhật sao cho đến khi kết thúc thì Jack trở lại ô xuất phát và số ô đất được đi qua là lớn nhất. Biết rằng Jack có K bình nước tăng lực và để di chuyển giữa 2 ô đất thì Jack cần tiêu tốn một số lượng bình năng lượng như sau:  
- Nếu ô đất sau cao hơn ô đất trước, Jack cần dùng số bình bằng với: (chênh lệch chiều cao)\*(chênh lệch chiều cao)\*2 +1   
- Nếu ô đất sau thấp hơn ô đất trước, Jack cần dùng số bình bằng với: (chênh lệch chiều cao)\*(chênh lệch chiều cao) +1  
- Nếu ô đất sau cao bằng ô đất trước, Jack chỉ cần dùng 1 bình năng lượng.  
Ví dụ: Giả sử cho khu đất rộng 4x4, và số bình năng lượng mà Jack có là 28. Xét trường hợp Jack xuất phát từ ô đất có độ cao là 3 như hình dưới.  
  
  
  
  
  
Năng lương mà Jack cần tiêu tốn là: 1+3+10+9+2+3=28. Như vậy Jack có thể hoàn thành lộ trình này. Số ô đất được đi qua trong trường hợp này là 6.  
Trong trường hợp Jack chạy bộ theo chiều ngược lại, số năng lượng mà Jack có sẽ không thể giúp anh chạy hết toàn bộ lộ trình này.  
  
  
  
  
  
Số bình năng lượng Jack cần trong trường hợp này là: 2+3+5+19+2+1=32.  
Hãy viết chương trình tìm ra lộ trình chạy mà Jack có thể đi qua nhiều ô đất nhất.   
[Constraints]  
- Lộ trình hợp lệ bắt buộc phải là hình chữ nhật (hoặc vuông).  
- Bộ input đưa vào luôn luôn có tuyến đường phù hợp với yêu cầu bài toán.  
[Input]  
- Đầu bài cho T test cases.  
- Dòng đầu tiên của mỗi test case bao gồm 2 số N và K.  
2<=N<=10  
K<=2000  
- N dòng tiếp theo, mỗi dòng chứa N giá trị, là chiều cao của các ô đất tương ứng.  
5  
4 28  
2 5 4 1  
2 1 3 3  
1 5 4 2  
1 3 1 3  
5 107  
10 6 6 3 9  
8 6 2 2 6  
8 8 4 2 9  
1 3 10 9 1  
8 2 6 4 7  
6 230  
9 2 3 2 5 1  
4 7 7 6 4 2  
8 7 9 2 4 3  
5 6 8 1 7 9  
9 6 1 8 7 8  
3 7 5 8 5 10  
8 311  
4 9 9 10 11 2 14 3  
14 13 3 4 6 3 3 6  
6 14 9 12 2 14 12 4  
2 12 12 11 11 7 4 15  
13 6 15 12 13 6 3 6  
9 7 5 12 11 1 10 1  
11 3 12 7 14 8 3 5  
9 2 13 8 5 4 6 4  
10 1920  
8 7 11 8 8 2 2 5 14 4  
19 13 14 4 13 2 1 16 16 4  
6 6 4 17 15 1 2 16 2 6  
4 16 12 8 8 13 5 7 12 5  
14 12 16 12 12 4 12 8 18 8  
8 10 12 15 10 1 20 18 3 13  
17 16 12 3 11 12 10 19 14 8  
16 9 15 11 12 9 11 6 16 17  
1 10 6 13 18 9 6 13 6 8  
15 1 2 12 13 16 8 12 6 5  
  
[Output]  
- Với mỗi test case, in ra số ô đất lớn nhất mà Jack có thể đi qua trong một lộ trình hợp lệ.  
#1 6  
#2 10  
#3 18  
#4 14  
#5 34



**else** **if** (huong != i && *Vs*[i] == 0 && huong + 1 != i && huong - 1 != i)

Mục đích chỗ +1 vs -1 đấy là để tránh quay lại hướng cũ

package luyende;  
  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
/\* false testcase thu 3. Vừa xong thì fix lỗi arrayindexoutof, giờ thì lỗi logic  
\* ý tưởng: for hết mảng mảng 2 chiều rồi backTrack trong đó tại 1 đỉnh bất kỳ rồi tìm ra số đỉnh đi được nhiều nhất  
\*/  
public class Jogging {  
Scanner sc = new Scanner([System.in](http://system.in/));  
int t, n, k, st, en, output;  
int map[][];  
  
void init() {  
output = 0;  
map = new int[n][n];  
}  
  
int NangLuongCan(int x, int y, int dai, int rong) {  
int check = 0;  
// sang phai  
int cnt = rong;  
while (cnt != 0) {  
y++;  
if (map[x][y] > map[x][y - 1]) {  
check += (map[x][y] - map[x][y - 1]) \* (map[x][y] - map[x][y - 1]) \* 2 + 1;  
} else if (map[x][y] < map[x][y - 1]) {  
check += (map[x][y] - map[x][y - 1]) \* (map[x][y] - map[x][y - 1]) + 1;  
} else {  
check += 1;  
}  
cnt--;  
}  
// xuong  
cnt = dai;  
while (cnt != 0) {  
x++;  
if (map[x][y] > map[x - 1][y]) {  
check += (map[x][y] - map[x - 1][y]) \* (map[x][y] - map[x - 1][y]) \* 2 + 1;  
} else if (map[x][y] < map[x - 1][y]) {  
check += (map[x][y] - map[x - 1][y]) \* (map[x][y] - map[x - 1][y]) + 1;  
} else {  
check += 1;  
}  
cnt--;  
}  
// sang trai  
cnt = rong;  
while (cnt != 0) {  
y--;  
if (map[x][y] > map[x][y + 1]) {  
check += (map[x][y] - map[x][y + 1]) \* (map[x][y] - map[x][y + 1]) \* 2 + 1;  
} else if (map[x][y] < map[x][y + 1]) {  
check += (map[x][y] - map[x][y + 1]) \* (map[x][y] - map[x][y + 1]) + 1;  
} else {  
check += 1;  
}  
cnt--;  
}  
// len  
cnt = dai;  
while (cnt != 0) {  
x--;  
// mình vừa bị sai logic ở chỗ if chỗ này (chỗ tọa độ) còn  
// arrindexoutof thì bị chỗ + -  
if (map[x][y] > map[x + 1][y]) {  
check += (map[x][y] - map[x + 1][y]) \* (map[x][y] - map[x + 1][y]) \* 2 + 1;  
} else if (map[x][y] < map[x + 1][y]) {  
check += (map[x][y] - map[x + 1][y]) \* (map[x][y] - map[x + 1][y]) + 1;  
} else {  
check += 1;  
}  
cnt--;  
}  
return check;  
  
}  
  
int NangLuongNguoc(int x, int y, int dai, int rong) {  
int check = 0;  
  
int cnt = rong;  
// xuong  
cnt = dai;  
while (cnt != 0) {  
x++;  
if (map[x][y] > map[x - 1][y]) {  
check += (map[x][y] - map[x - 1][y]) \* (map[x][y] - map[x - 1][y]) \* 2 + 1;  
} else if (map[x][y] < map[x - 1][y]) {  
check += (map[x][y] - map[x - 1][y]) \* (map[x][y] - map[x - 1][y]) + 1;  
} else {  
check += 1;  
}  
cnt--;  
}  
// sang phai  
cnt = rong;  
while (cnt != 0) {  
y++;  
if (map[x][y] > map[x][y - 1]) {  
check += (map[x][y] - map[x][y - 1]) \* (map[x][y] - map[x][y - 1]) \* 2 + 1;  
} else if (map[x][y] < map[x][y - 1]) {  
check += (map[x][y] - map[x][y - 1]) \* (map[x][y] - map[x][y - 1]) + 1;  
} else {  
check += 1;  
}  
cnt--;  
}  
// len  
cnt = dai;  
while (cnt != 0) {  
x--;  
// mình vừa bị sai logic ở chỗ if chỗ này (chỗ tọa độ) còn  
// arrindexoutof thì bị chỗ + -  
if (map[x][y] > map[x + 1][y]) {  
check += (map[x][y] - map[x + 1][y]) \* (map[x][y] - map[x + 1][y]) \* 2 + 1;  
} else if (map[x][y] < map[x + 1][y]) {  
check += (map[x][y] - map[x + 1][y]) \* (map[x][y] - map[x + 1][y]) + 1;  
} else {  
check += 1;  
}  
cnt--;  
}  
// sang trai  
cnt = rong;  
while (cnt != 0) {  
y--;  
if (map[x][y] > map[x][y + 1]) {  
check += (map[x][y] - map[x][y + 1]) \* (map[x][y] - map[x][y + 1]) \* 2 + 1;  
} else if (map[x][y] < map[x][y + 1]) {  
check += (map[x][y] - map[x][y + 1]) \* (map[x][y] - map[x][y + 1]) + 1;  
} else {  
check += 1;  
}  
cnt--;  
}  
  
return check;  
}  
  
void solution() {  
t = sc.nextInt();  
for (int tc = 1; tc <= t; tc++) {  
n = sc.nextInt();  
k = sc.nextInt();  
init();  
for (int i = 0; i < n; i++) {  
for (int j = 0; j < n; j++) {  
map[i][j] = sc.nextInt();  
}  
}  
for (int i = 0; i < n - 1; i++) {  
for (int j = 0; j < n - 1; j++) {  
int dai, rong;  
for (dai = 1; dai < n; dai++) {  
for (rong = 1; rong < n; rong++) {  
// ko để (i + dai <= n && j + rong <= n) được vì giả sử chạy từ (0,1) thì chạy max tới n-1, (độ dài hìn

h chữ nhật max nhất là n (từ 0 tới (n-1) độ dài là n)  
if (i + dai < n && j + rong < n) {  
int ans;  
ans = NangLuongCan(i, j, dai, rong);  
if (ans <= k) {  
if ((dai + rong) \* 2 > output) {  
output = (dai + rong) \* 2;  
}  
}  
ans = NangLuongNguoc(i, j, dai, rong);  
if (ans <= k) {  
if ((dai + rong) \* 2 > output) {  
output = (dai + rong) \* 2;  
}  
}  
}  
}  
}  
}  
}  
System.out.println("#" + tc + " " + output);  
}  
}  
  
public static void main(String[] args) throws Exception {  
System.setIn(new FileInputStream("Jogging.txt"));  
Jogging j = new Jogging();  
j.solution();  
}  
  
}

50  
5 74  
1 15 2 6 16   
20 12 10 8 14   
8 8 6 16 20   
19 8 17 12 18   
6 2 15 15 14   
5 151  
5 8 20 14 15   
15 2 18 7 5   
10 17 3 5 4   
19 5 7 13 20   
8 20 10 9 15   
5 156  
12 1 2 4 2   
1 8 20 4 15   
10 3 4 1 18   
11 10 13 5 14   
11 13 13 3 4   
5 32  
1 20 16 5 7   
13 20 18 16 4   
11 19 4 3 12   
3 11 16 9 4   
3 8 15 20 13   
5 134  
4 18 10 4 7   
6 4 14 9 6   
14 6 3 18 20   
15 4 14 11 3   
10 10 1 7 20   
5 83  
11 15 4 6 12   
15 4 17 1 10   
10 17 11 15 20   
20 13 10 14 10   
7 2 7 13 12   
5 149  
14 19 12 12 11   
11 9 16 13 17   
4 17 10 3 4   
12 5 14 8 15   
19 11 14 20 14   
5 48  
3 10 9 19 6   
18 17 20 10 12   
17 4 19 15 9   
11 11 6 17 18   
18 11 9 2 7   
5 86  
11 10 7 3 9   
12 7 8 3 10   
9 9 6 13 17   
3 11 5 10 9   
20 20 3 7 15   
5 98  
19 14 18 14 10   
10 14 18 2 9   
1 3 2 10 2   
11 7 1 9 19   
9 10 16 13 7   
5 170  
12 6 1 3 7   
4 4 20 13 15   
2 19 18 18 13   
14 9 14 1 18   
13 8 14 8 17   
5 168  
13 1 2 13 4   
8 16 1 3 8   
3 2 20 7 9   
17 6 14 13 6   
14 10 20 19 11   
5 164  
5 12 18 11 6   
15 10 3 5 4   
5 10 8 13 5   
1 18 19 10 17   
1 1 5 13 15   
6 175  
18 19 4 5 18 4   
18 3 15 19 4 7   
20 18 2 8 17 17   
17 2 19 5 19 9   
13 11 15 15 6 1   
3 12 15 9 7 17   
6 125  
7 3 3 15 5 11   
4 7 1 11 1 4   
18 10 9 11 11 16   
2 1 14 11 1 16   
7 18 18 4 1 19   
17 13 13 1 6 17   
7 1905  
1 6 11 6 13 4 13   
16 10 20 9 5 2 5   
10 20 8 5 10 3 3   
2 6 15 9 9 20 10   
15 2 2 11 8 17 12   
19 13 7 2 20 1 6   
2 5 4 9 5 15 18   
7 920  
11 9 19 4 20 6 6   
10 18 5 2 12 2 16   
2 7 16 3 20 17 12   
12 17 17 4 11 2 6   
7 5 8 1 2 5 13   
10 5 17 2 20 12 1   
17 16 20 5 13 16 5   
7 1985  
15 12 6 2 14 20 13   
4 9 13 19 20 9 19   
14 10 5 1 9 11 5   
16 13 14 16 14 15 4   
16 6 16 16 15 13 12   
17 9 2 20 7 14 20   
18 14 15 10 1 10 7   
7 1628  
20 12 19 20 20 1 10   
6 4 1 7 13 4 3   
17 2 2 4 3 10 3   
6 1 9 17 15 12 18   
12 13 5 15 10 11 7   
9 17 4 7 7 14 14   
12 14 1 19 8 17 12   
7 1279  
16 14 14 13 12 6 7   
16 6 16 20 6 11 14   
19 17 20 14 6 19 4   
2 8 7 12 18 6 18   
3 4 16 9 20 7 7   
1 1 19 14 8 7 7   
3 11 16 5 13 3 7   
7 1160  
9 15 20 17 20 18 6   
9 11 14 18 13 11 17   
7 3 19 6 1 5 12   
9 7 8 20 10 11 3   
12 9 8 4 3 14 17   
17 7 6 9 16 3 17   
2 13 1 17 9 8 7   
7 554  
5 9 20 3 16 13 7   
5 3 7 8 11 3 19   
17 8 3 5 8 3 5   
7 12 16 4 1 19 12   
4 1 9 15 11 14 10   
20 12 2 19 6 6 20   
18 4 7 17 3 18 12   
7 1372  
20 1 9 20 6 8 1   
1 2 9 15 15 9 6   
16 20 6 6 20 10 14   
19 7 8 4 14 10 5   
20 17 14 17 5 1 19   
10 7 20 6 9 3 10   
17 13 3 11 1 5 3   
7 1928  
20 8 4 11 18 18 5   
4 16 7 16 7 2 9   
10 1 17 13 2 10 1   
13 18 14 19 8 11 15   
13 20 15 2 16 1 8   
3 10 14 13 19 7 19   
20 7 11 5 13 16 10   
7 1018  
1 2 17 12 15 15 19   
20 4 12 9 4 8 7   
3 4 20 13 16 7 9   
10 12 2 20 9 9 3   
12 14 7 2 1 19 18   
7 15 3 17 1 16 3   
16 4 2 8 12 8 1   
7 1054  
18 6 14 2 7 7 3   
8 12 5 5 2 19 15   
4 3 4 13 12 13 10   
20 11 4 12 20 13 12   
1 8 14 10 16 5 15   
18 16 4 14 2 5 18   
5 18 7 13 13 6 2   
7 1426  
18 14 8 3 19 20 4   
18 14 7 12 15 19 11   
6 20 19 13 20 2 1   
13 1 5 2 9 17 6   
13 2 9 12 8 3 5   
20 13 20 15 4 18 13   
12 12 9 8 7 17 17   
7 662  
16 20 19 17 20 3 20   
19 8 3 17 15 19 18   
6 10 1 7 12 7 4   
9 4 5 11 16 11 11   
17 5 3 1 13 13 2   
19 20 12 3 4 5 14   
5 17 3 14 10 12 9   
8 1818  
1 15 18 18 10 12 11 8   
14 14 13 10 9 2 10 1   
2 4 12 9 6 2 6 15   
2 17 19 3 9 1 17 14   
7 4 19 5 8 14 14 19   
3 2 18 12 15 19 12 8   
7 2 4 16 5 9 17 7   
15 1 18 20 17 7 5 10   
8 1171  
11 2 14 1 6 12 9 14   
5 18 17 17 1 20 1 10   
8 18 6 18 8 2 4 15   
17 11 20 8 6 7 12 19   
19 9 1 16 14 10 9 14   
9 11 8 3 1 6 7 9   
3 14 7 10 7 16 19 11   
10 1 10 7 10 9 6 6   
8 1055  
15 17 9 6 7 14 19 14   
4 18 15 6 17 11 3 19   
5 11 12 1 4 5 7 19   
12 11 11 17 3 11 6 10   
10 8 2 17 16 3 5 5   
2 19 5 13 17 8 4 7   
4 8 4 14 17 5 15 13   
11 5 8 6 10 5 14 9   
8 1110  
8 12 4 17 1 9 9 2   
19 14 9 1 1 15 12 17   
18 5 17 17 18 2 9 18   
20 20 12 2 10 6 6 15   
15 4 16 2 5 7 12 7   
15 9 3 9 1 16 11 7   
12 1 19 19 6 4 14 17   
12 8 15 7 16 20 10 10   
8 1760  
7 9 11 16 7 18 6 4   
4 14 4 20 6 13 9 7   
4 12 4 18 19 10 9 14   
17 16 2 14 10 8 14 3   
10 20 1 17 9 2 11 19   
12 5 13 4 6 8 12 16   
4 11 1 17 12 5 7 11   
3 15 1 19 10 15 17 2   
8 1608  
15 14 18 10 10 19 8 2   
19 16 8 2 18 12 20 13   
6 10 17 13 6 14 3 2   
15 14 1 3 17 9 18 8   
10 16 18 17 10 13 8 3   
10 11 12 2 8 12 14 13   
13 20 17 18 10 18 12 18   
18 19 20 13 6 2 20 19   
8 1899  
9 16 14 14 10 15 1 3   
6 9 11 13 17 11 7 2   
11 7 13 10 12

11 7 9   
7 2 20 3 6 17 19 18   
5 19 12 4 10 12 14 16   
3 19 17 13 4 3 13 19   
17 20 19 1 14 17 17 17   
4 5 5 20 12 19 8 1   
8 1803  
10 5 6 12 5 20 1 15   
7 4 6 7 16 13 6 1   
6 10 7 18 17 11 19 19   
11 9 13 6 9 2 14 7   
2 18 2 7 3 6 19 2   
4 17 20 7 5 14 19 3   
16 13 15 12 4 2 9 7   
7 11 7 9 12 8 10 10   
9 1428  
17 18 3 1 15 10 15 1 10   
4 14 18 3 1 14 15 8 7   
12 9 16 2 4 20 4 5 17   
7 14 14 10 8 12 9 19 10   
15 5 2 5 11 17 12 12 2   
16 15 20 3 20 11 11 10 5   
14 20 18 13 16 3 20 4 4   
11 19 15 15 10 14 8 12 10   
1 11 6 1 19 17 8 7 8   
9 994  
9 10 10 5 13 4 19 19 3   
19 9 2 2 17 4 9 15 4   
3 10 12 14 13 5 3 17 8   
16 7 14 15 16 16 7 8 16   
8 14 2 14 1 14 5 13 3   
19 4 16 8 15 4 16 14 9   
8 11 7 10 1 8 5 19 4   
1 5 8 14 10 12 13 15 1   
9 9 10 13 9 16 11 14 16   
9 1589  
1 3 1 13 10 7 13 10 1   
2 11 6 14 18 15 18 19 19   
19 19 18 1 12 1 20 14 5   
2 9 6 2 19 8 12 18 1   
2 1 15 20 2 8 3 10 6   
14 9 13 1 2 3 9 14 12   
1 7 2 7 1 19 10 8 14   
19 17 6 12 3 1 13 2 15   
18 6 14 7 15 5 4 9 6   
9 1049  
3 5 6 8 4 16 3 1 1   
9 1 4 3 12 3 5 12 4   
7 13 19 17 5 15 11 3 19   
16 16 12 8 17 4 11 4 10   
20 9 16 8 9 12 6 5 12   
19 20 15 20 7 19 9 19 15   
5 7 18 17 10 20 17 10 1   
8 9 10 11 8 6 13 15 12   
14 5 19 3 3 13 8 17 4   
9 1349  
8 16 13 6 5 19 17 16 8   
16 14 18 11 2 20 20 8 5   
20 17 19 13 13 4 20 12 16   
11 12 7 11 17 16 6 1 13   
7 10 15 8 4 2 12 10 1   
10 3 2 7 17 3 3 12 6   
6 20 11 12 11 16 13 1 17   
5 4 16 11 1 4 19 6 15   
5 11 8 18 5 7 7 4 16   
9 1402  
9 13 2 5 16 10 16 4 14   
16 7 13 19 16 6 13 15 7   
6 3 9 4 14 12 9 3 2   
2 11 2 10 7 5 8 1 8   
3 13 18 19 20 16 17 5 14   
12 13 8 1 2 5 6 2 19   
18 19 20 4 4 15 13 9 10   
7 10 3 11 8 12 4 10 17   
1 15 12 14 3 19 15 4 6   
9 1550  
5 3 4 14 7 11 4 11 6   
1 11 11 10 4 19 20 13 8   
16 18 1 7 17 19 7 19 8   
12 10 12 5 20 1 16 6 8   
14 14 5 10 17 11 3 4 5   
16 1 5 8 4 2 15 4 18   
10 1 7 9 12 11 9 11 18   
20 1 10 18 7 6 3 13 17   
2 12 2 6 8 14 9 7 4   
10 1737  
10 14 10 7 15 1 11 12 11 19   
14 2 1 8 12 13 8 19 19 4   
20 13 6 19 17 18 11 12 16 12   
4 5 18 8 10 2 15 16 19 9   
14 13 5 19 8 7 8 3 1 6   
3 18 4 20 7 3 14 17 19 15   
12 7 19 16 10 2 20 7 7 1   
1 3 16 18 13 16 12 19 6 14   
17 3 5 12 10 3 20 10 13 5   
8 14 12 11 15 6 2 2 18 6   
10 931  
18 6 10 7 17 12 9 5 4 16   
20 5 12 16 17 6 2 3 16 17   
11 2 12 8 17 13 20 16 20 12   
5 1 10 3 13 1 1 20 4 17   
11 11 9 12 13 19 4 5 8 2   
1 18 17 19 14 15 1 20 14 7   
13 1 17 11 16 12 10 8 3 16   
13 11 10 1 5 19 4 3 9 4   
18 4 14 14 16 12 7 6 4 1   
8 10 15 13 9 9 4 8 13 8   
10 1906  
16 9 20 15 15 5 8 16 1 18   
8 9 20 4 11 2 6 14 14 18   
20 16 6 18 1 19 3 10 11 12   
1 16 11 17 15 2 13 16 16 12   
6 11 13 7 16 20 17 16 8 8   
3 11 19 12 2 7 5 18 8 9   
16 8 11 9 14 15 5 3 11 20   
3 6 9 15 13 19 18 16 11 13   
5 15 11 11 10 4 12 15 10 5   
2 9 10 19 12 12 11 19 12 9   
10 563  
11 19 5 11 13 6 8 14 20 11   
9 13 4 20 15 4 3 7 9 11   
13 15 20 10 5 15 8 13 12 16   
4 3 19 15 7 18 3 8 3 9   
14 3 9 17 3 19 9 7 19 20   
14 8 18 14 9 4 1 17 1 8   
3 4 9 20 7 9 8 5 5 8   
15 19 5 7 20 6 13 16 11 14   
16 14 11 5 1 6 9 17 5 20   
8 1 6 20 5 2 3 12 9 16   
10 1903  
6 11 5 5 17 11 1 13 19 7   
9 19 17 4 4 18 20 4 2 3   
18 16 18 8 15 11 19 20 7 12   
17 14 3 15 7 9 19 20 13 8   
18 14 1 12 19 2 19 10 9 9   
2 7 1 13 9 5 2 2 20 5   
18 8 4 14 18 3 13 15 6 20   
10 1 7 2 6 12 20 7 18 20   
5 15 7 5 16 8 2 13 5 7   
12 10 1 16 8 3 6 6 5 9   
10 1401  
5 6 8 9 5 9 9 10 9 20   
19 4 14 1 4 19 2 7 5 17   
15 20 13 19 5 14 10 4 16 17   
17 10 17 17 1 6 14 20 12 16   
20 13 1 20 6 17 8 17 5 8   
7 1 1 3 10 8 4 15 9 6   
18 19 3 14 8 13 16 11 17 6   
6 16 14 13 1 7 20 16 20 3   
5 4 3 17 17 3 1 11 14 8   
4 9 12 11 18 6 3 17 9 10   
10 1182  
7 13 19 16 14 15 10 14 3 18   
16 1 2 15 4 11 12 1 9 6   
12 2 14 11 10 2 9 12 6 11   
18 6 7 3 16 19 18 17 18 16   
14 15 6 20 6 2 13 18 15 10   
13 20 8 20 9 20 15 13 7 11   
7 1 13 17 7 19 11 15 11 12   
4 11 12 16 2 6 10 12 2 18   
6 9 14 10 5 18 4 17 18 3   
5 9 6 11 10 10 10 5 5 13

#1 4  
#2 4  
#3 8  
#4 0  
#5 4  
#6 4  
#7 6  
#8 0  
#9 8  
#10 4  
#11 6  
#12 4  
#13 4  
#14 6  
#15 6  
#16 22  
#17 14  
#18 24  
#19 18  
#20 18  
#21 20  
#22 12  
#23 20  
#24 20  
#25 18  
#26 18  
#27 22  
#28 14  
#29 24  
#30 20  
#31 24  
#32 24  
#33 24  
#34 24  
#35 26  
#36 26  
#37 24  
#38 22  
#39 22  
#40 22  
#41 24  
#42 24  
#43 28  
#44 24  
#45 22  
#46 32  
#47 14  
#48 26  
#49 28  
#50 30

package luyende;  
  
import java.io.FileInputStream;  
import java.util.Scanner;  
  
public class ThamKhaoJogging {  
static int N, K, max;  
static int[][] Arr;  
static int[] Dx = { 1, -1, 0, 0 };  
static int[] Dy = { 0, 0, 1, -1 };  
static int x1, y1;  
static int[] Vs = new int[4];  
static int[][] VsM;  
  
public static void main(String args[]) throws Exception {  
System.setIn(new FileInputStream("Jogging.txt"));  
Scanner sc = new Scanner([System.in](http://system.in/));  
int T = sc.nextInt();  
for (int test = 1; test <= T; test++) {  
N = sc.nextInt();  
K = sc.nextInt();  
Arr = new int[N][N];  
VsM = new int[N][N];  
for (int i = 0; i < N; i++) {  
for (int j = 0; j < N; j++) {  
Arr[i][j] = sc.nextInt();  
}  
}  
max = 0;  
int huong = -1;  
for (int i = 0; i < N; i++) {  
for (int j = 0; j < N; j++) {  
x1 = i;  
y1 = j;  
BT(i, j, 0, huong, 0);  
}  
}  
System.out.println("#" + test + " " + max);  
}  
}  
  
static void BT(int x, int y, int count, int huong, int total) {  
if (x == x1 && y == y1 && total != 0) {  
if (count > max) {  
max = count;  
}  
} else {  
for (int i = 0; i < 4; i++) {  
int a = x + Dx[i];  
int b = y + Dy[i];  
if (a >= 0 && a <= N - 1 && b >= 0 && b <= N - 1 && VsM[a][b] == 0) {  
int d = Arr[x][y] - Arr[a][b];  
if (huong == i) {  
if (d >= 0 && total + d \* d + 1 <= K) {  
VsM[a][b] = 1;  
BT(a, b, count + 1, i, total + d \* d + 1);  
VsM[a][b] = 0;  
}  
if (d < 0 && total + d \* d \* 2 + 1 <= K) {  
VsM[a][b] = 1;  
BT(a, b, count + 1, i, total + d \* d \* 2 + 1);  
VsM[a][b] = 0;  
}  
} else if (huong != i && Vs[i] == 0 && huong + 1 != i && huong - 1 != i) {  
Vs[i] = 1;  
if (d >= 0 && total + d \* d + 1 <= K) {  
VsM[a][b] = 1;  
BT(a, b, count + 1, i, total + d \* d + 1);  
VsM[a][b] = 0;  
}  
if (d < 0 && total + d \* d \* 2 + 1 <= K) {  
VsM[a][b] = 1;  
BT(a, b, count + 1, i, total + d \* d \* 2 + 1);  
VsM[a][b] = 0;  
}  
Vs[i] = 0;  
}  
}  
}  
}  
}  
}

ko có && huong + 1 != i && huong - 1 != i thì false case 4