جمع ماتریس

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: متوسط
- طراح: امیر محمد گنجی زاده

شما یک ماتریس در اختیار دارید که هر عنصر آن یک عدد صحیح است.

شما باید به ازای همه ی اعداد از ۰ تا k بگید که چند سطر یا ستون یا قطر(خط های اریب)، مجموعی برابر با این عدد دارن.

ورودی

در خط اول به ترتیب n و m و k ورودی داده میشه :

$$0 \leq n, m, k \leq 100$$

و پس از آن در n خط و در هر خط m عدد ورودی داده میشود.

خروجی

خروجی شامل **k + 1** خطه که در هر خط جواب مربوط به عدد اون خطه.

ورودی نمونه

```
2 3 4
1 1 1
2 2 2
```

خروجی نمونه

```
0
2
2
8
0
```

توضیح نمونه

قطر (اریب) ها و مجموع آنها

```
----------------------
##*          *##
###  = 1     ###  = 1
----------------------
#*#          #*#
##*  = 3     *##  = 3
----------------------
*##          ##*
#*#  = 3     #*#  = 3
----------------------
###          ###
*##  = 2     ##*  = 2
----------------------
```

اتصال پایگاه داده

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح : متوسط
- طراح: آریا صدیق

پروژه اولیه رو از این لینک دانلود کنید.

آریا یکی از توسعه‌دهندگان تیم **DataConnect** است که مشغول طراحی سیستمی برای مدیریت پایگاه‌های داده مختلف می‌باشد. این سیستم قرار است چندین پایگاه داده را مدیریت کند که ممکن است هرکدام از آن‌ها نیاز به ایجاد اتصال‌های مختلف داشته باشند.

آریا به‌سرعت متوجه می‌شود که در این سیستم، چندین اتصال همزمان به پایگاه داده برقرار می‌شود. این موضوع باعث بروز مشکلاتی از جمله کاهش کارایی، مصرف زیاد منابع، و حتی احتمال بروز خطا در فرآیند ارتباط با پایگاه داده می‌شود. با توجه به اهمیت بهینه‌سازی منابع و حفظ کارایی سیستم، تصمیم می‌گیرد که به جای ایجاد اتصالات متعدد، تنها یک اتصال به پایگاه داده در طول اجرای برنامه وجود داشته باشد. این روش می‌تواند علاوه بر بهبود عملکرد، مدیریت منابع سیستم را نیز ساده‌تر کند.

بنابراین، تصمیم می‌گیرد که این سیستم را با استفاده از **الگوی Singleton** طراحی کند. در این طراحی، تنها یک نمونه از کلاس اتصال به پایگاه داده در طول زمان وجود خواهد داشت و هر زمان که نیاز به اتصال به پایگاه داده باشد، از همان اتصال استفاده خواهد شد.

وظیفه شما:

شما باید یک کلاس به نام **DatabaseConnection** طراحی کنید که ویژگی‌های زیر را داشته باشد:

۱. **تنها یک اتصال فعال وجود دارد:**
   - در ابتدا که اولین درخواست برای اتصال داده می‌شود، اتصال برقرار شده و برای همه درخواست‌های بعدی از همان اتصال استفاده می‌شود.
   - برای جلوگیری از ایجاد اتصالات اضافی، تنها یک نمونه از این کلاس در طول اجرای برنامه وجود خواهد داشت.

۲. **متدهای کلاس DatabaseConnection:**

- **connect()**: متدی که تنها یک بار اتصال برقرار می‌کند و در صورت اتصال موفق مقدار `true` را برمی‌گرداند. اگر اتصال قبلاً برقرار شده باشد، مقدار `false` باید برگردانده شود.
- **disconnect()**: متدی که اتصال برقرار شده را قطع می‌کند و مقدار `true` را برمی‌گرداند. در صورتی که هیچ اتصال فعالی وجود نداشته باشد، مقدار `false` باید برگردانده شود.
- **getConnectionStatus()**: متدی که وضعیت اتصال را با یک مقدار Boolean نمایش می‌دهد. در صورت برقراری اتصال مقدار `true` و در غیر این صورت مقدار `false` باید برگردانده شود.
- **getInstance()**: این تابع باید در صورت وجود اتصال دیتابیس آن را برگرداند و در غیر این صورت، نمونه‌ای ساخته و برگرداند.

راهنمایی‌ها:

- **الگوی Singleton** باید به گونه‌ای پیاده‌سازی شود که هیچ‌گاه بیش از یک اتصال به پایگاه داده ایجاد نشود.
- به منظور جلوگیری از مشکلات همزمانی و وضعیت‌های پیش‌بینی‌نشده در محیط‌های چندنخی، پیاده‌سازی شما باید به گونه‌ای باشد که هنگام درخواست اتصال، همیشه یک نمونه واحد از کلاس ایجاد شود.

مثال

ورودی نمونه

```
public class DatabaseConnection {
public static void main(String[] args) {
        // تست برقراری اتصال
DatabaseConnection connection1 = DatabaseConnection.getInstance(
System.out.println(connection1.connect());
        // تست تلاش دوباره برای برقراری اتصال
DatabaseConnection connection2 = DatabaseConnection.getInstance(
System.out.println(connection2.connect());
        // تست وضعیت اتصال
System.out.println(connection1.getConnectionStatus());
        // تست قطع اتصال
```

```
System.out.println(connection1.disconnect());
        // تست وضعیت پس از قطع اتصال
System.out.println(connection1.getConnectionStatus());
    }}
```

خروجی نمونه

```
true
false
true
true
false
```

شما باید یک فایل Zip شامل یک فایل به نام `DatabaseConnection.java` را آپلود کنید.

بوکینگ دات کام

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: متوسط
- طراحان: امیررضا یزدان پناه و زهرا عزیزی

شما یک نقشه از نمای بالای یک هتل پیدا کرده اید. در این نقشه دیوارهای هتل با # مشخص شده و فضای اتاق ها با * نمایش داده شده اند. شما می خواهید تعداد اتاق های این هتل را پیدا کنید. از آنجایی که خیلی به جاوا علاقه مندید، تصمیم گرفتید یک کد جاوا برای انجام این کار بنویسید!

▼ راهنمایی

از الگوریتم DFS استفاده کنید. یکی از کاربردهای این الگوریتم، پیدا کردن مولفه همبندی گراف است. چگونه می توانید اتاق های هتل را به مولفه های یک گراف مرتبط سازید؟

▼ DFS

الگوریتم جستجوی عمق اول (DFS)

اینجا کمی با الگوریتم جستجوی عمق اول (Depth First Search یا DFS) آشنا می‌شویم. این الگوریتم یکی از روش‌های پایه‌ای پیمایش گراف است.

کاربردها

- **درخت‌های تصمیم‌گیری**: برای بررسی تمامی مسیرهای ممکن در فضای تصمیم و انتخاب بهترین گزینه.
- **کاوش در گراف‌ها**: یافتن مؤلفه‌های همبند، کشف چرخه‌ها، مسیر یابی، و موارد بسیاری که نیاز به گردش عمیق در ساختار گراف دارند.

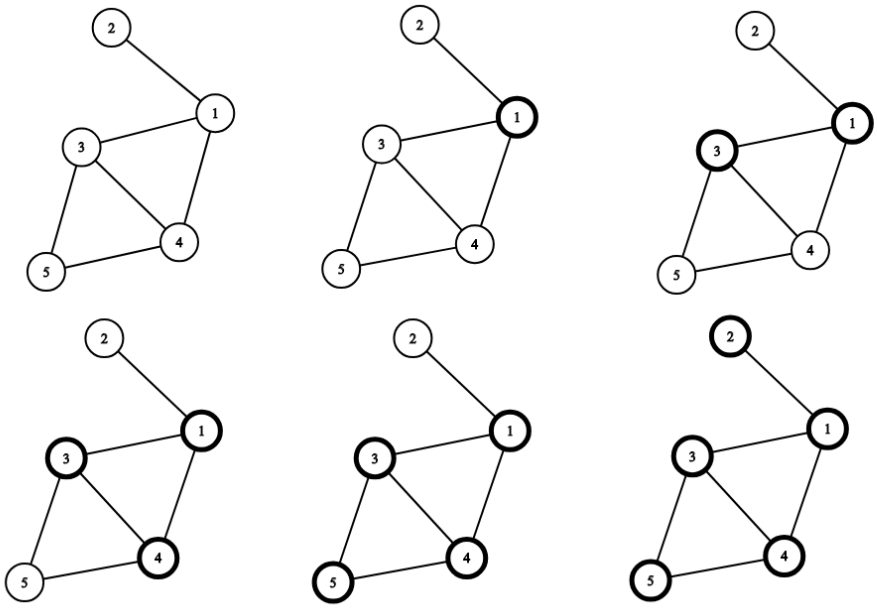
توضیح الگوریتم

الگوریتم به‌صورت بازگشتی کار می‌کند. ابتدا همهٔ راس‌ها «سفید» (بازدیدنشده) در نظر گرفته می‌شوند. هنگامی که DFS روی راسی `v` فراخوانی شود:

۱. `v` را «خاکستری» (در حال پردازش) علامت می‌زنیم.
۲. برای هر همسایهٔ `u` از `v` که هنوز سفید است، بازگشتی DFS روی `u` اجرا می‌کنیم.
۳. پس از اتمام تمام همسایه‌ها، `v` «مشکی» (پایان پردازش) می‌شود.

این روند باعث می‌شود که الگوریتم تا ژرفای هر مسیر در گراف پیش برود و سپس به عقب بازگردد.

شکل زیر نمونه یک `dfs` با شروع از راس 1 است.



به این صورت که ابتدا `dfs` برای 1 صدا زده شده است، سپس برای 3، سپس برای 4، سپس برای 5 و بعد در نهایت 2، آخرین اجرای `dfs` بوده است.

پیچیدگی زمانی

- هر راس حداکثر یک‌بار وارد می‌شود (وقتی که سفید است).
- برای هر راس، مرور همسایه‌ها مجموعاً به اندازهٔ درجهٔ آن راس هزینه دارد.

نتیجتاً پیچیدگی کلی:

```
O(n + m)
```

که `n` تعداد رئوس و `m` تعداد یال‌ها است.

ارتباط الگوریتم DFS با مسأله شمارش اتاق‌های هتل

در این سوال، نقشهٔ هتل به صورت ماتریسی از کاراکترهای `#` (دیوار) و `*` (اتاق) نمایش داده می‌شود. برای به‌کارگیری DFS:

۱. **تبدیل ماتریس به گراف ضمنی**: هر سلول `*` یک راس فرض می‌شود و هر جابجایی در جهت‌های بالا/پایین/چپ/راست معادل یک یال بین راس‌ها است.

۲. **پیمایش مؤلفه‌های همبندی**: با شروع یک فراخوانی DFS از هر سلولی که هنوز بازدید نشده باشد، تمام سلول‌های متعلق به همان اتاق در یک مؤلفهٔ همبندی قرار می‌گیرند و علامت‌گذاری می‌شوند.

۳. **شمارش اتاق‌ها**: هر بار که یک فراخوانی DFS جدید شروع شود، یک اتاق جدید شناسایی شده است. در نهایت تعداد این فراخوانی‌ها برابر است با تعداد اتاق‌ها.

بدین ترتیب، الگوریتم DFS به سرعت و با پیچیدگی $O(n \cdot m)$ (تعداد سلول‌ها) می‌تواند تعداد مؤلفه‌های متصل (اتاق‌ها) را در نقشهٔ هتل محاسبه کند.

سودو کد برای متود DFS:

```
function DFS(x, y):
    mark visited[x][y] = true
    for each (dx, dy) in [(-1,0), (1,0), (0,-1), (0,1)]:
        nx = x + dx
        ny = y + dy
        if (nx, ny) is inside boundaries and cell (nx, ny) is ope
            DFS(nx, ny)
```

ورودی

در خط اول ورودی دو عدد صحیح `n` و `m` که نشان‌دهنده اندازه نقشه هتل هستند به شما داده می شوند.

$$6 \leq n, m \leq 20$$

- نکته: بعد از ورودی گرفتن `n` و `m` ، یک بار با استفاده از `scanner.nextLine();` به خط بعدی رفته و سپس ورودی کاراکتر ها را بگیرید.

در n خط بعدی در هر خط m کارکتر که `#` یا `*` هستند به شما داده می شود. تضمین می شود که اتاق ها به صورت قطری با یکدیگر ارتباط **ندارند** و اطراف هتل لزوما دیوار است. (با `#` پوشیده شده است.)

منظور از ارتباط قطری اتاق ها، مانند نمونه زیر است:

```
######
##*###
###*##
######
```

خروجی

در تنها خط خروجی، تعداد اتاق های هتل را چاپ کنید.

مثال

ورودی نمونه ۱

```
6 9
#########
#**#####*#
#**#####*#
#########
###***###
#########
```

خروجی نمونه ۱

```
3
```

ورودی نمونه ۲

```
17 11
###########
######*####
##***#*##*#
#########*#
#*****#####
#######*#*#
##*##*#*###
##*########
```

```
#######*###
#*#####*#*#
#*#######*#
###***#####
#######*#*#
#*###*#####
###*#####**#
##########
##########
```

خروجی نمونه ۲

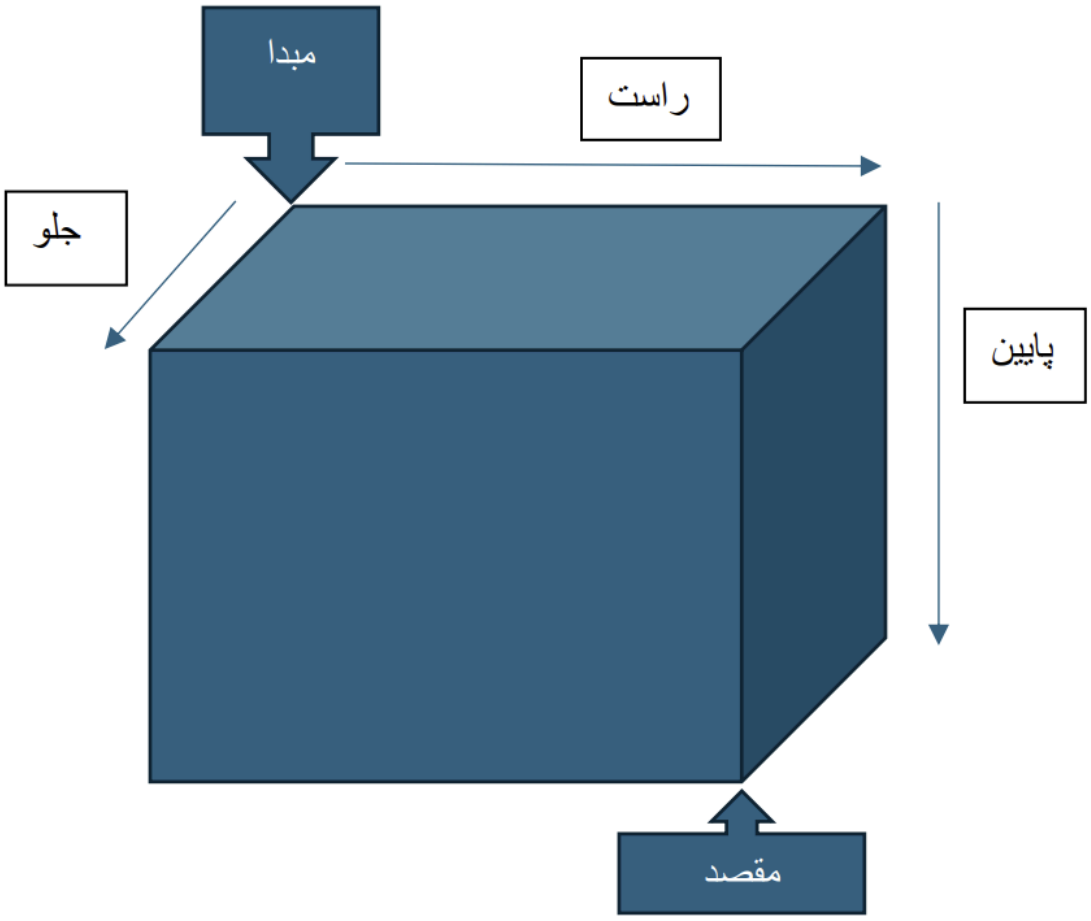
18

شجاع

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: متوسط
- طراح: نیما سلطانی

دنیای شجاع معکبی است و هر راس دارای یک امتیاز خاص است.اون در گوشه بالا سمت چپ است و میخواهد به گوشه پایین سمت راست برود.به او کمک کنید مسیری را پیدا کند که **بیشترین امتیاز** را داشته

باشد(چون او شجاع است فقط حرکت به راست،پایین و جلو انجام میدهد و بعد از عبور از هر راس،امتیاز آن راس را دریافت میکند)



ورودی

در خط اول سه عدد که به ترتیب نمایش دهنده عمق(تعداد لایه ها) و تعداد سطر های هر لایه و تعداد ستون های هر لایه در دنیای شجاع است میاید و در سطر های بعدی متناسب با دنیای شجاع،امتیاز ها ذکر میشوند(به مثال ها توجه کنید)

خروجی

امتیاز نهایی شجاع را چاپ کنید

مثال

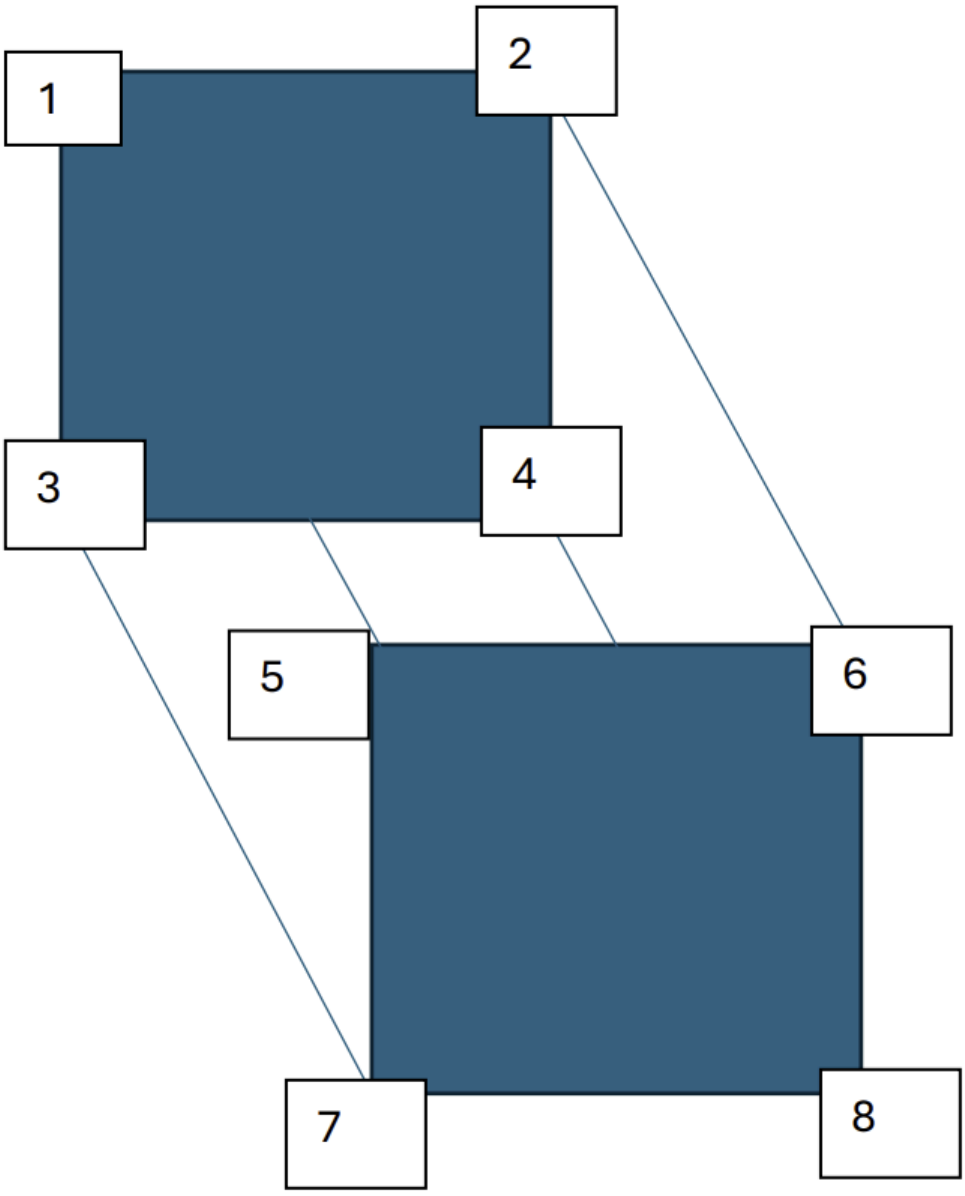
ورودی نمونه ۱

```
2 2 2
1 2
3 4
5 6
7 8
```

خروجی نمونه ۱

```
21
```

شجاع ابتدا در خانه ای است که مقدار 1 دارد.سپس به ترتیب به خانه ای با مقدار5 و 7 و8 میرود.



ورودی نمونه ۲

```
2 3 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

```
17 18 19 20
21 22 23 24
```

خروجی نمونه ۲

```
121
```

بازی با ماتریس

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: سخت
- طراح: مهدی کریمی

ورودی

در خط اول ورودی به شما به ترتیب تعداد سطر ها و ستون های یک ماتریس مربعی داده می شود. سپس در خط های بعدی ماتریس مدنظر به شما داده خواهد شد.

خروجی

در صورتی که ماتریس داده شده، با ترانهاده آن، برابر باشد، ابتدا ماتریس را 180 درجه در جهت عقربه های ساعت دوران دهید، سپس ماتریس را به صورت ستونی پیمایش کنید. درغیر این صورت ابتدا ماتریس را 90 درجه در جهت عقربه های ساعت دوران دهید، و سپس آن را به صورت مارپیچی پیمایش کنید و درایه های آن را چاپ کنید.

محدودیت ها

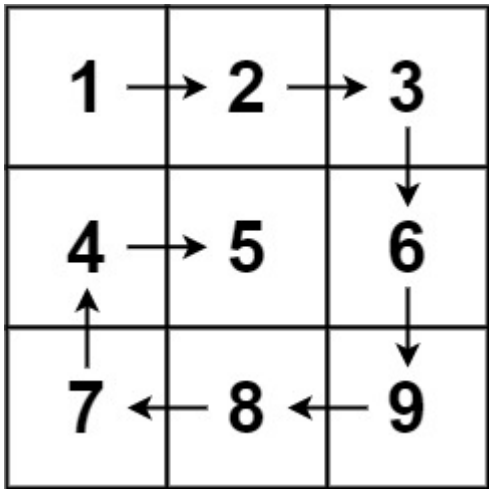
$$m = matrix.length$$

$$n = matrix[i].length$$

$$1 \leq m, n \leq 30$$

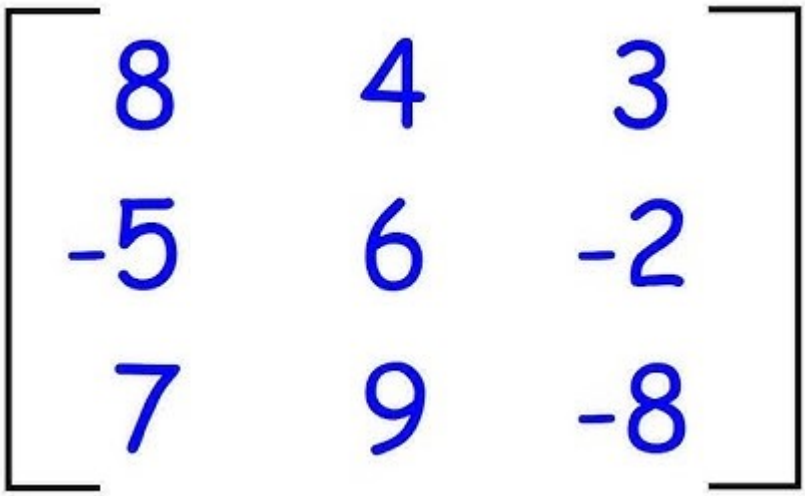
$$-1000 \leq matrix[i][j] \leq 1000$$

مثال از پیمایش مارپیچی:



در نتیجه ترتیب اعضا به صورت 1 2 3 6 9 8 7 4 5 می باشد.

مثال از پیمایش ستونی:



در صورت پیمایش ستونی ترتیب اعضا به صورت 8- 2- 3 9 6 4 7 5- 8 می باشد.

ورودی نمونه 1

```
3 3
1 2 3
2 4 5
3 5 6
```

خروجی نمونه 1

```
6 5 3 5 4 2 3 2 1
```

▼ توضیح

چون ماتریس داده شده متقارن است، پس ابتدا 180 درجه دوران می دهیم و سپس درایه های آن را به صورت ستونی چاپ می کنیم.

ورودی نمونه 2

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

خروجی نمونه 2

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

▼ توضیح

از آنجایی که ماتریس داده شده متقارن نیست، ابتدا آن را 90 درجه می چرخانیم و سپس درایه های آن را چاپ می کنیم.