Problem Solving (C47)

هذا البرنامج التدريبي مُصاغ بعناية لتمكين المتدربين من تطوير قدراتهم الفكرية على غرار المبرمجين المحترفين، والتعاون بكفاءة ضمن فريق محترف في شركة "جمال تك" أو أي مؤسسة متعددة الجنسيات أخرى. نظرًا لأهمية اللغة الإنجليزية في بيئة العمل العالمية، يتم تقديم المحتوى التدريبي بالإنجليزية. لا يشترط إتقان اللغة بشكل كامل، لكن من الضروري امتلاك القدرة الكافية لفهم المتطلبات وتنفيذها بشكل فعّال. يُمكن للمتدربين استخدام مترجم جوجل أو الاستعانة بـ "شات جي بي تي" للتغلب على أية عقبات لغوية، المهم هو الفهم الدقيق للمطلوب وتحقيقه بنجاح.

لتعظيم الاستفادة من التدريب، يُنصح بمحاولة حل التمارين بشكل مستقل لمدة ساعة واحدة على الأقل قبل الرجوع إلى الحل المرفق في نهاية الملف.

قد يتضمن الحل كودًا برمجيًا غير مفسر بعد، والغرض من ذلك هو تشجيعك على محاولة فهم الأكواد البرمجية الجديدة التي لم تتعرض لها من قبل. هذه المهارة ضرورية في سوق العمل، حيث تتطور لغات البرمجة باستمرار ويظهر كل يوم لغات جديدة. ستواجه دائمًا أكوادًا لم تدرسها من قبل، ومن المهم أن تكون قادرًا على فهمها بنفسك دون الحاجة إلى دراسة مسبقة. يمكنك الاستعانة بمحرك البحث جوجل، أو استخدام ChatGPT، أو حتى اللجوء لأصدقائك للمساعدة. الهدف الأساسي هو أن تصل إلى فهم معنى كل كود بأي طريقة ممكنة لتتمكن من إيجاد موقعك في سوق العمل.

إن وجود كود برمجي غير مفسر يشكل تحديًا يتوجب عليك إيجاد حل له. هذا النوع من التدريبات يعد جزءًا أساسيًا من تدريبات 'Problem Solving'، التي تهدف إلى تمكينك من أداء عملك بفاعلية بغض النظر عن التحديات والعقبات. هذه القدرة على حل المشكلات هي ما يتمتع به العاملون في 'جمال تك'، ومن الضروري أن تطور في نفسك هذه المهارة لتصبح عضوًا فعالًا في فريق عمل 'جمال تك'.

Gammal Tech's Optimized Workspace Layout

Background:
Gammal Tech, a vanguard in the software industry, renowned for its innovative approaches and advanced office facilities, is embarking on a project to optimize its office space. The aim is to create an environment that fosters collaboration and maximizes productivity while ensuring comfort for its team members.

Problem Description:
Your task is to help Gammal Tech design an optimized layout for their new office floor. The floor is represented as a 2D grid, where each cell can be a workstation, a relaxation area, or a path. The goal is to maximize the accessibility of relaxation areas from every workstation while ensuring a minimum distance for noise reduction.

Constraints:

- The office floor is a 2D grid of size $M \times N$ ($1 \leq M, N \leq 100$).
- Each cell in the grid can be:
    - 0: representing a path.

- 1: representing a workstation.
- 2: representing a relaxation area.

- There must be at least one relaxation area and one workstation.
- A path (0) is required to access any cell.

Input Format:

- The first line contains two space-separated integers M and N, the dimensions of the floor.

- M lines follow, each containing N integers describing the floor layout.

Output Format:
Output a single integer, the maximum distance from any workstation to the nearest relaxation area, minimized across the entire layout.

Sample Input:

```
5 5
1 0 0 0 2
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
2 0 0 0 1
```

Sample Output:

```
3
```

Explanation:

In the given layout, the maximum distance from any workstation to the nearest relaxation area is 3. This distance is found by calculating the shortest path from each workstation to the nearest relaxation area. The algorithm ensures that all workstations have optimal access to relaxation areas, showcasing Gammal Tech's commitment to a balanced and efficient work environment.

لتحقيق أقصى فائدة من التدريب، يُوصى ببذل محاولة مستقلة لحل التمارين لمدة لا تقل عن ساعة واحدة. تجنب الاطلاع على بقية الملف حتى تكمل عملية التفكير في الحل. بعد ذلك، جرب حلك بنفسك على المدخلات الموضحة. إذا واجهت مدخلات لم تتوقعها، فهذا يعد فرصة لتطوير مهارة جديدة ضمن مسيرتك التعليمية. المهندس المحترف يجب أن يضمن أن برنامجه يعمل مع جميع أنواع المدخلات، وهذه مهارة يتم تطويرها عبر التجربة والخطأ. لذا، من الضروري ألا تطلع على المدخلات المتوقعة قبل أن تجرب الحل بنفسك. هذه هي الطريقة الأمثل لتنمية هذه المهارة.

بعد اختبار المدخلات المقترحة، إذا كانت النتائج تختلف عما هو مدون في الملف، فيُنصح بمحاولة حل التمرين مرة أخرى لمدة ساعة على الأقل قبل الرجوع إلى الحل الموجود في نهاية الملف.

Test Case 1: Minimum Size Input

Input

```
1 1
1
```

Expected Output

```
0
```

Explanation:
There is only one cell which is a workstation. Since there are no relaxation areas, the distance is 0 by default.

Test Case 2: Relaxation Area in Every Corner

Input

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

Expected Output

```
2
```

Explanation:

The workstation is surrounded by relaxation areas in all four corners. The maximum distance to the nearest relaxation area is 2.

Test Case 3: Workstation Surrounded by Paths

Input

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

Expected Output

```
1
```

Explanation:

The workstation is surrounded by paths, with a relaxation area just one cell away.

Test Case 4: Long Narrow Office

Input

```
1 5
1 0 0 0 2
```

Expected Output

```
4
```

Explanation:
The office is a long, narrow corridor with a workstation at one end and a relaxation area at the other end. The distance is 4.

Test Case 5: Multiple Workstations with Varying Distances

Input

```
3 4
1 0 0 2
0 0 0 0
1 0 0 0
```

Expected Output

```
3
```

Explanation:

There are two workstations at different distances from the relaxation area. The maximum distance is 3, from the workstation in the bottom left corner.

لتحقيق أقصى استفادة من التدريب، من المستحسن أن تخصص وقتًا إضافيًا - لا يقل عن ساعة - لمحاولة حل التمرين مرة أخرى بمفردك قبل الرجوع إلى الحل المقترح. هذه العملية المتكررة من التجربة والخطأ تعتبر استراتيجية فعالة في تعزيز مهاراتك البرمجية وتعميق فهمك للمفاهيم. تذكر أن التحدي والمثابرة هما المفتاحان للتطور في مجال البرمجة.

C Programming Solution:

To address this challenge, we will employ a Breadth-First Search (BFS) algorithm. This approach will enable us to efficiently identify the shortest routes connecting each workstation to the closest relaxation zone. Subsequently, we will ascertain the longest of these shortest paths. If this concept seems complex at first, don't be concerned. Consider it a preliminary exercise for the upcoming course on data structures and algorithms. By engaging with this material now, you'll be laying a strong foundation for a more profound understanding of these topics in the future.

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define MAX 100

typedef struct {
    int x, y;
} Point;

int office[MAX][MAX];
int visited[MAX][MAX];

void initVisited(int m, int n) {
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            visited[i][j] = 0;
}

int isValid(int x, int y, int m, int n) {
    return x >= 0 && y >= 0 && x < m && y < n;
}

int bfs(int startX, int startY, int m, int n) {
    int dx[] = {1, -1, 0, 0};
    int dy[] = {0, 0, 1, -1};

    initVisited(m, n);
    Point queue[MAX * MAX];
    int front = 0, rear = 0;

    queue[rear++] = (Point){startX, startY};
    visited[startX][startY] = 1;

    while (front < rear) {
        Point p = queue[front++];
        for (int i = 0; i < 4; i++) {
            int newX = p.x + dx[i];
            int newY = p.y + dy[i];
```

```
            if (isValid(newX, newY, m, n) && !visited[newX][newY] &&
office[newX][newY] != 1) {
                visited[newX][newY] = visited[p.x][p.y] + 1;
                if (office[newX][newY] == 2)
                    return visited[newX][newY] - 1;
                queue[rear++] = (Point){newX, newY};
            }
        }
    }
    return INT_MAX;
}

int main() {
    int m, n;
    scanf("%d %d", &m, &n);

    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &office[i][j]);

    int maxDist = 0;
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (office[i][j] == 1) {
                int dist = bfs(i, j, m, n);
                if (dist > maxDist)
                    maxDist = dist;
            }
        }
    }

    printf("%d\n", maxDist);
    return 0;
}
```