

[LintCode领扣题解 \(/problem\)](#) / 骑士的最短路径II · Knight Shortest Path II

骑士的最短路径II · Knight Shortest Path II

中文

[亚马逊 \(/problem/?tags=amazon\)](#)[Breadth-first Search \(/problem/?tags=breadth-first-search\)](#)[动态规划 \(/problem/?tags=dynamic-programming\)](#)

描述

在一个 $n * m$ 的棋盘中(二维矩阵中 0 表示空 1 表示有障碍物), 骑士的初始位置是 $(0, 0)$, 他想要达到 $(n - 1, m - 1)$ 这个位置, 骑士只能从左边走到右边。找出骑士到目标位置所需要走的最短路径并返回其长度, 如果骑士无法达到则返回 -1 。

样例

例1:

输入:
[[0,0,0,0],[0,0,0,0],[0,0,0,0]]
输出:
3
解释:
[0,0]->[2,1]->[0,2]->[2,3]

例2:

输入:
[[0,1,0],[0,0,1],[0,0,0]]
输出:
-1

说明

如果骑士所在位置为 (x,y) , 那么他的下一步一步到达以下位置:

$(x + 1, y + 2)$
 $(x - 1, y + 2)$
 $(x + 2, y + 1)$
 $(x - 2, y + 1)$

在线评测地址: <https://www.lintcode.com/problem/knight-shortest-path-ii/> (<https://www.lintcode.com/problem/knight-shortest-path-ii/>)

[收起题目描述 ^](#)

语言类型

[ALL \(25\)](#)[java \(11\)](#)[python \(10\)](#)[cpp \(4\)](#)[上传题解](#)

令狐冲

更新于 8/24/2020, 10:36:47 AM

简单动态规划。每个点都有至多4个前缀点, 直接求最小值转移即可。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param grid a chessboard included 0 and 1
     * @return the shortest path
     */
    public int shortestPath2(boolean[][] grid) {
        // Write your code here
        int n = grid.length;
        if (n == 0)
            return -1;
        int m = grid[0].length;
        if (m == 0)
            return -1;

        int[][] f = new int[n][m];
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < m; ++j)
                f[i][j] = Integer.MAX_VALUE;

        f[0][0] = 0;
        for (int j = 1; j < m; ++j)
            for (int i = 0; i < n; ++i)
                if (!grid[i][j]) {
                    if (i >= 1 && j >= 2 && f[i - 1][j - 2] != Integer.MAX_VALUE)
                        f[i][j] = Math.min(f[i][j], f[i - 1][j - 2] + 1);
                    if (i + 1 < n && j >= 2 && f[i + 1][j - 2] != Integer.MAX_VALUE)
                        f[i][j] = Math.min(f[i][j], f[i + 1][j - 2] + 1);
                    if (i >= 2 && j >= 1 && f[i - 2][j - 1] != Integer.MAX_VALUE)
                        f[i][j] = Math.min(f[i][j], f[i - 2][j - 1] + 1);
                    if (i + 2 < n && j >= 1 && f[i + 2][j - 1] != Integer.MAX_VALUE)
                        f[i][j] = Math.min(f[i][j], f[i + 2][j - 1] + 1);
                }

        if (f[n - 1][m - 1] == Integer.MAX_VALUE)
            return -1;

        return f[n - 1][m - 1];
    }
}
```

👍 获赞 3

💬 4 条评论

你的口袋题库

2000+算法真题、国内外名企题库免费开放



九章算法APP

令狐冲

更新于 7/22/2020, 6:32:33 PM

简单动态规划。每个点都有至多4个前缀点, 直接求最小值转移即可。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param grid a chessboard included 0 and 1
     * @return the shortest path
     */
    int shortestPath2(vector<vector<bool>>& grid) {
        // Write your code here
        int n = grid.size();
        if (n == 0)
            return -1;
        int m = grid[0].size();
        if (m == 0)
            return -1;

        vector<vector<int>> f(n, vector<int>(m, INT_MAX));
        f[0][0] = 0;
        for (int j = 1; j < m; ++j)
            for (int i = 0; i < n; ++i)
                if (!grid[i][j]) {
                    if (i >= 1 && j >= 2 && f[i - 1][j - 2] != INT_MAX)
                        f[i][j] = min(f[i][j], f[i - 1][j - 2] + 1);
                    if (i + 1 < n && j >= 2 && f[i + 1][j - 2] != INT_MAX)
                        f[i][j] = min(f[i][j], f[i + 1][j - 2] + 1);
                    if (i >= 2 && j >= 1 && f[i - 2][j - 1] != INT_MAX)
                        f[i][j] = min(f[i][j], f[i - 2][j - 1] + 1);
                    if (i + 2 < n && j >= 1 && f[i + 2][j - 1] != INT_MAX)
                        f[i][j] = min(f[i][j], f[i + 2][j - 1] + 1);
                }

        if (f[n - 1][m - 1] == INT_MAX)
            return -1;

        return f[n - 1][m - 1];
    }
};
```

👍 获赞 3

💬 添加评论



令狐冲

更新于 11/1/2020, 11:09:47 AM

滚动数组的版本

python

java

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
DIRECTIONS = [
    (-1, -2),
    (1, -2),
    (-2, -1),
    (2, -1),
]

class Solution:
    # @param {boolean[][]} grid a chessboard included 0 and 1
    # @return {int} the shortest path
    def shortestPath2(self, grid):
        if not grid or not grid[0]:
            return -1

        n, m = len(grid), len(grid[0])

        # state: dp[i][j % 3] 代表从 0,0 跳到 i,j 的最少步数
        dp = [[float('inf')] * 3 for _ in range(n)]

        # initialize: 0,0 是起点
        dp[0][0] = 0

        # function
        for j in range(1, m):
            for i in range(n):
                dp[i][j % 3] = float('inf')
                if grid[i][j]:
                    continue
                for delta_x, delta_y in DIRECTIONS:
                    x, y = i + delta_x, j + delta_y
                    if 0 <= x < n and 0 <= y < m:
                        dp[i][j % 3] = min(dp[i][j % 3], dp[x][y % 3] + 1)

        # answer
        if dp[n - 1][(m - 1) % 3] == float('inf'):
            return -1
        return dp[n - 1][(m - 1) % 3]
```

👍 获赞 2 💬 3 条评论



九章-加贺

更新于 7/26/2020, 1:44:39 AM

使用双向宽度优先搜索算法, 效率+++

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
```

```
*/
class Point{
    int x,y;
    public Point(int x,int y) {
        this.x = x;
        this.y = y;
    }
}

public class Solution {
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    int[][] FORWARD_DIRECTIONS = {
        {1, 2},
        {-1, 2},
        {2, 1},
        {-2, 1}
    };
    int[][] BACKWARD_DIRECTIONS = {
        {-1, -2},
        {1, -2},
        {-2, -1},
        {2, -1}
    };

    public int shortestPath2(boolean[][] grid) {
        if (grid == null || grid.length == 0) {
            return -1;
        }
        if (grid[0] == null || grid[0].length == 0) {
            return -1;
        }
        int n = grid.length;
        int m = grid[0].length;
        if (grid[n - 1][m - 1] == true) {
            return -1;
        }
        if (n * m == 1) {
            return 0;
        }

        Queue<Point> forwardQueue = new LinkedList<Point>();
        Queue<Point> backwardQueue = new LinkedList<Point>();
        boolean[][] forwardSet = new boolean[n][m];
        boolean[][] backwardSet = new boolean[n][m];

        forwardQueue.offer(new Point(0, 0));
        backwardQueue.offer(new Point(n - 1, m - 1));
        forwardSet[0][0] = true;
        backwardSet[n - 1][m - 1] = true;

        int distance = 0;
        while (!forwardQueue.isEmpty() && !backwardQueue.isEmpty()) {
            distance++;
            if (extendQueue(forwardQueue, FORWARD_DIRECTIONS, forwardSet, backwardSet, grid)) {
                return distance;
            }

            distance++;
            if (extendQueue(backwardQueue, BACKWARD_DIRECTIONS, backwardSet, forwardSet, grid)) {
                return distance;
            }
        }

        return -1;
    }
}
```

```
boolean extendQueue(Queue<Point> queue,
                    int[][] directions,
                    boolean[][] visited,
                    boolean[][] oppositeVisited,
                    boolean[][] grid) {
    int queueLength = queue.size();
    for (int i = 0; i < queueLength; i++) {
        Point head = queue.poll();
        int x = head.x;
        int y = head.y;

        for (int j = 0; j < 4; j++) {
            int newX = x + directions[j][0];
            int newY = y + directions[j][1];
            if (!isValid(newX, newY, grid, visited)) {
                continue;
            }
            if (oppositeVisited[newX][newY] == true) {
                return true;
            }

            queue.offer(new Point(newX, newY));
            visited[newX][newY] = true;
        }
    }

    return false;
}

boolean isValid(int x,
                int y,
                boolean[][] grid,
                boolean[][] visited) {
    if (x < 0 || x >= grid.length) {
        return false;
    }
    if (y < 0 || y >= grid[0].length) {
        return false;
    }
    if (grid[x][y]) {
        return false;
    }
    if (visited[x][y]) {
        return false;
    }

    return true;
}
```

👍 获赞 2 💬 添加评论



令狐冲

更新于 6/22/2020, 9:34:37 PM

使用了坐标变换数组的代码

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
DIRECTIONS = [
    (-1, -2),
    (1, -2),
    (-2, -1),
    (2, -1),
]

class Solution:
    # @param {boolean[][]} grid a chessboard included 0 and 1
    # @return {int} the shortest path
    def shortestPath2(self, grid):
        if not grid or not grid[0]:
            return -1

        n, m = len(grid), len(grid[0])

        # state: f[i][j] 代表从 0,0 跳到 i,j 的最少步数
        f = [[sys.maxsize for j in range(m)] for _ in range(n)]

        # initialize: f[0][0] 是起点
        f[0][0] = 0

        # function
        for j in range(m):
            for i in range(n):
                if grid[i][j]:
                    continue
                for delta_x, delta_y in DIRECTIONS:
                    x, y = i + delta_x, j + delta_y
                    if 0 <= x < n and 0 <= y < m:
                        f[i][j] = min(f[i][j], f[x][y] + 1)

        if f[n - 1][m - 1] == sys.maxsize:
            return -1

        return f[n - 1][m - 1]

```

👍 获赞 2

💬 3 条评论



九章管理员

更新于 12/9/2020, 11:23:12 AM

使用双向宽度优先搜索算法, 效率+++

python

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课

```

```
* - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
*/
FORWARD_DIRECTIONS = (
    (1, 2),
    (-1, 2),
    (2, 1),
    (-2, 1),
)

BACKWARD_DIRECTIONS = (
    (-1, -2),
    (1, -2),
    (-2, -1),
    (2, -1),
)

class Solution:
    def shortestPath2(self, grid):
        if not grid or not grid[0]:
            return -1

        n, m = len(grid), len(grid[0])
        if grid[n - 1][m - 1]:
            return -1
        if n * m == 1:
            return 0

        forward_queue = collections.deque([(0, 0)])
        forward_set = set([(0, 0)])
        backward_queue = collections.deque([(n - 1, m - 1)])
        backward_set = set([(n - 1, m - 1)])

        distance = 0
        while forward_queue and backward_queue:
            distance += 1
            if self.extend_queue(forward_queue, FORWARD_DIRECTIONS, forward_set, backward_set, grid):
                return distance

            distance += 1
            if self.extend_queue(backward_queue, BACKWARD_DIRECTIONS, backward_set, forward_set, grid):
                return distance

        return -1

    def extend_queue(self, queue, directions, visited, opposite_visited, grid):
        for _ in range(len(queue)):
            x, y = queue.popleft()
            for dx, dy in directions:
                new_x, new_y = (x + dx, y + dy)
                if not self.is_valid(new_x, new_y, grid, visited):
                    continue
                if (new_x, new_y) in opposite_visited:
                    return True
                queue.append((new_x, new_y))
                visited.add((new_x, new_y))

        return False

    def is_valid(self, x, y, grid, visited):
        if x < 0 or x >= len(grid):
            return False
        if y < 0 or y >= len(grid[0]):
            return False
        if grid[x][y]:
            return False
        if (x, y) in visited:
            return False
        return True
```


👍 获赞 1 💬 添加评论



令狐冲

更新于 6/9/2020, 7:03:58 AM

简单动态规划。每个点都有至多4个前缀点, 直接求最小值转移即可。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    # @param {boolean[][]} grid a chessboard included 0 and 1
    # @return {int} the shortest path
    def shortestPath2(self, grid):
        # Write your code here
        n = len(grid)
        if n == 0:
            return -1

        m = len(grid[0])
        if m == 0:
            return -1

        f = [ [sys.maxsize for j in range(m)] for _ in range(n)]

        f[0][0] = 0
        for j in range(m):
            for i in range(n):
                if not grid[i][j]:
                    if i >= 1 and j >= 2 and f[i - 1][j - 2] != sys.maxsize:
                        f[i][j] = min(f[i][j], f[i - 1][j - 2] + 1)
                    if i + 1 < n and j >= 2 and f[i + 1][j - 2] != sys.maxsize:
                        f[i][j] = min(f[i][j], f[i + 1][j - 2] + 1)
                    if i >= 2 and j >= 1 and f[i - 2][j - 1] != sys.maxsize:
                        f[i][j] = min(f[i][j], f[i - 2][j - 1] + 1)
                    if i + 2 < n and j >= 1 and f[i + 2][j - 1] != sys.maxsize:
                        f[i][j] = min(f[i][j], f[i + 2][j - 1] + 1)

        if f[n - 1][m - 1] == sys.maxsize:
            return -1

        return f[n - 1][m - 1]
```

👍 获赞 1 💬 添加评论



Z同学

更新于 11/24/2020, 3:50:32 AM

根据 Knight Shortest Path 的解法, 用BFS。区别在于: 这一题需要自己写Point class,需要定义source和destination, 可走的步数变成了4步, 其它都一样。注意: 在 Knight Shortest Path 中, 规定了destination一定可到达 (不能是barrier), 但在本题 Knight Shortest Path II中并没有规定一定可以到达! 所以一定要先check 边界 && check 是否可到达之后再 check answer, 或者在程序一开始check destination可到达!

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
```

* - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code

```
*/
```

```
class Point
```

```
{
    int x, y;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

```
public class Solution {
```

```
/**
```

```
 * @param grid: a chessboard included 0 and 1
```

```
 * @return: the shortest path
```

```
*/
```

```
public int shortestPath2(boolean[][] grid) {
```

```
    int R = grid.length - 1;
```

```
    int C = grid[0].length - 1;
```

```
    int[] dirRow = {1, -1, 2, -2};
```

```
    int[] dirCol = {2, 2, 1, 1};
```

```
    Point source = new Point(0, 0);
```

```
    Point destination = new Point(R, C);
```

```
    Queue<Point> queue = new LinkedList<>();
```

```
    queue.offer(source);
```

```
    /*
```

```
    // In this problem, using hashSet take extra time
```

```
    // so mark the grid as barrier to avoid duplicates!
```

```
    HashSet<Point> set = new HashSet<>();
```

```
    set.add(source);
```

```
    */
```

```
    int steps = 0;
```

```
    while (!queue.isEmpty())
```

```
    {
```

```
        steps++;
```

```
        int currSize = queue.size();
```

```
        for (int i = 0; i < currSize; i++)
```

```
        {
```

```
            Point point = queue.poll();
```

```
            for (int j = 0; j < dirRow.length; j++)
```

```
            {
```

```
                Point position = new Point(point.x + dirRow[j],
                                             point.y + dirCol[j]);
```

```
                if (!inbound(position, R, C) || isBarrier(position, grid))
```

```
                {
```

```
                    continue;
```

```
                }
```

```
                // check answer:
```

```
                if (position.x == destination.x && position.y == destination.y)
```

```
                {
```

```
                    return steps;
```

```
                }
```


```
                queue.offer(position);
```

```
        // mark position as barrier to avoid duplicates!!
        grid[position.x][position.y] = true;
    }
}

return -1;
}

private boolean inbound(Point point, int R, int C)
{
    if (point.x >= 0 && point.x <= R && point.y >= 0 && point.y <= C)
    {
        return true;
    }
    return false;
}

private boolean isBarrier(Point point, boolean[][] grid)
{
    if (grid[point.x][point.y])
    {
        return true;
    }
    return false;
}
}
```

 获赞 3 添加评论**zjchen**

更新于 10/14/2020, 11:21:11 PM

BFS解法, 按步搜索, 如果终点值为1, 直接返回-1, 否则从左向右搜索, 已经访问过的点置为1. 和DP解法相比, 不需要额外空间。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
def shortestPath2(self, grid):
    # write your code here
    if not grid or grid[-1][-1] == 1:
        return -1

    n = len(grid)
    m = len(grid[0])

    delta = [(1,2), (-1,2), (2,1), (-2,1)]
    queue = [(0,0)]
    step = 0
    while queue:
        size = len(queue)
        step += 1
        for i in range(size):
            x, y = queue.pop()
            for delta_x, delta_y in delta:
                if x + delta_x == n - 1 and y + delta_y == m - 1:
                    return step
                if n > x + delta_x >= 0 and m > y + delta_y >= 0 and grid[x + delta_x][y + delta_y] != 1:
                    grid[x + delta_x][y + delta_y] = 1
                    queue.insert(0, (x+delta_x, y+delta_y))

    return -1
```

👍 获赞 3

💬 1 条评论



九章用户N2FZTL

更新于 6/9/2020, 7:03:48 AM

为了通过DP统计当前位置的值, 需要知道之前4个位置dpi-1 (), dpi+1 (), dpi-2 (), dpi+2 ()的值, 这些点都在当前点的左边, 因此需要按“列”遍历。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    public int shortestPath2(boolean[][] grid) {
        if(grid == null || grid.length == 0) return -1;
        int m = grid.length;
        int n = grid[0].length;
        int[][] dp = new int[m+4][n+4];
        dp[2][2] = 1;
        for(int j=2; j<n+2; j++){
            for(int i=2; i<m+2; i++){
                if(!(i==2 && j==2) && !grid[i-2][j-2]){
                    int min = Integer.MAX_VALUE;
                    min = dp[i-1][j-2] == 0? min:Math.min(min, dp[i-1][j-2]);
                    min = dp[i+1][j-2] == 0? min:Math.min(min, dp[i+1][j-2]);
                    min = dp[i-2][j-1] == 0? min:Math.min(min, dp[i-2][j-1]);
                    min = dp[i+2][j-1] == 0? min:Math.min(min, dp[i+2][j-1]);

                    dp[i][j] = min == Integer.MAX_VALUE?0:min+1;
                }
            }
        }
        return dp[m+1][n+1]-1;
    }
}

```

👍 获赞 3

💬 2 条评论



S同学

更新于 11/24/2020, 3:50:42 AM

发一个bfs版本的, 把step放入自定义class Node 中。。。。。。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Node {
    int x;
    int y;
    int step;
    public Node(int x, int y, int step) {
        this.x = x;
        this.y = y;
        this.step = step;
    }
}

```

```
public class Solution {
    int[] xDirections = new int[]{1, -1, 2, -2};
    int[] yDirections = new int[]{2, 2, 1, 1};
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    public int shortestPath2(boolean[][] grid) {
        // write your code here
        Queue<Node> queue = new LinkedList<>();
        int m = grid.length, n = grid[0].length;

        Node start = new Node(0, 0, 0);
        queue.offer(start);

        while (!queue.isEmpty()) {
            Node node = queue.poll();

            if (node.x == m - 1 && node.y == n - 1) {
                return node.step;
            }

            for (int i = 0; i < 4; i++) {
                int x = node.x + xDirections[i];
                int y = node.y + yDirections[i];

                if (!isInbound(x, y, grid)) {
                    continue;
                }

                if (grid[x][y]) {
                    continue;
                }

                grid[x][y] = true;
                queue.offer(new Node(x, y, node.step + 1));
            }
        }

        return -1;
    }

    private boolean isInbound(int x, int y, boolean[][] grid) {
        if (x < 0 || x >= grid.length) {
            return false;
        }

        if (y < 0 || y >= grid[0].length) {
            return false;
        }

        return true;
    }
}
```

👍 获赞 2

💬 添加评论



明尼苏达大角羊

更新于 10/14/2020, 11:21:05 PM

循环数组的写法, 用O(3n)的额外空间 (1) 列遍历j必须放在外循环 (当然无论是否循环数组都得这样...) (2) 为了使用循环数组, 每一次更新时必须把当前step先初始化为maxsize, 覆盖旧数值, 避免错误 (3) 为了不覆盖掉原点, j必须从1开始 (当然第一列除了原点也不可能走到)

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def shortestPath2(self, grid):
        if not grid or not grid[0]:
            return -1

        n, m = len(grid), len(grid[0])
        if n == 0 or m == 0:
            return -1

        steps = [[sys.maxsize] * 3 for i in range(n)]
        steps[0][0] = 0

        for j in range(1, m):
            for i in range(n):
                steps[i][j % 3] = sys.maxsize
                if grid[i][j] == 1:
                    continue

                for delta_i, delta_j in [(1, 2), (-1, 2), (2, 1), (-2, 1)]:
                    pre_i, pre_j = i - delta_i, j - delta_j

                    if pre_i < 0 or pre_i >= n or pre_j < 0 or pre_j >= m or steps[pre_i][pre_j % 3] == sys.maxsize:
                        continue

                    steps[i][j % 3] = min(steps[i][j % 3], steps[pre_i][pre_j % 3] + 1)

        if steps[n - 1][(m - 1) % 3] == sys.maxsize:
            return -1

        return steps[n - 1][(m - 1) % 3]

```

👍 获赞 2 💬 添加评论



九章用户G1LOUJ

更新于 8/3/2020, 2:46:29 PM

硅谷求职算法集训营版本, 使用双向BFS优化搜索过程, 降低时间复杂度。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    public int shortestPath2(boolean[][] grid) {
        if (grid == null || grid.length == 0 || grid[0].length == 0) {

```

```
        return -1;
    }

    int m = grid.length, n = grid[0].length;
    if (grid[0][0] || grid[m - 1][n - 1]) {
        return -1;
    }

    if (m == 1 && n == 1) return 0;

    Queue<Integer> queA = new LinkedList<>();
    boolean[][] vA = new boolean[m][n];
    queA.offer(0);
    vA[0][0] = true;

    Queue<Integer> queB = new LinkedList<>();
    boolean[][] vB = new boolean[m][n];
    queB.offer(m * n - 1);
    vB[m - 1][n - 1] = true;

    Queue<Integer> que = null;
    boolean[][] vCur = null, vOp = null;

    int res = 0, sign = 0;

    int[] dirX = {-1, 1, -2, 2};
    int[] dirY = {2, 2, 1, 1};

    while (!queA.isEmpty() && !queB.isEmpty()) {
        if (queA.size() <= queB.size()) {
            que = queA;
            vCur = vA;
            vOp = vB;
            sign = 1;
        } else {
            que = queB;
            vCur = vB;
            vOp = vA;
            sign = -1;
        }

        res++;
        int l = que.size();

        while (l-- != 0) {
            int cur = que.poll();
            int x = cur / n, y = cur % n;

            for (int i = 0; i < 4; ++i) {
                int xx = x + sign * dirX[i], yy = y + sign * dirY[i];
                if (xx >= 0 && xx < m && yy >= 0 && yy < n && !grid[xx][yy]) {
                    if (vOp[xx][yy]) {
                        return res;
                    }

                    if (!vCur[xx][yy]) {
                        que.offer(xx * n + yy);
                        vCur[xx][yy] = true;
                    }
                }
            }
        }
    }

    return -1;
}
```


👍 获赞 2

💬 1 条评论

**Jet**

更新于 6/9/2020, 7:03:53 AM

骑士问题II。这一题需要注意的是for循环时,需先从纵坐标开始,再横坐标,这样才不会出现某个点f(i)之前的点未初始化的状态。(或者也可以先初始化?)。然后当要求最短路径时,初始化每个点的状态为INT_MAX,求max径时,初始化每个点的状态为0

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    int shortestPath2(vector<vector<bool>> &grid) {
        // write your code here
        if(grid.size()==0||grid[0].size()==0||grid[0][0]==1){
            return -1;
        }
        int n=grid.size(), m=grid[0].size();
        int f[n][m];
        f[0][0]=0;
        vector<int> dx={1,-1,2,-2};
        vector<int> dy={2,2,1,1};
        for(int j=0;j<m;j++){
            for(int i=0;i<n;i++){
                if(i==0&&j==0){
                    continue;
                }
                f[i][j]=INT_MAX;
                if(grid[i][j]){
                    continue;
                }
                for(int k=0;k<4;k++){
                    int x=i-dx[k];
                    int y=j-dy[k];
                    if(x<0||x>=n||y<0||y>=m){
                        continue;
                    }
                    if(f[x][y]==INT_MAX){
                        continue;
                    }
                    f[i][j]=min(f[i][j],f[x][y]+1);
                }
            }
        }
        if(f[n-1][m-1]==INT_MAX){
            return -1;
        }
        return f[n-1][m-1];
    }
};
```

👍 获赞 1

💬 添加评论

**九章用户PN3URB**

更新于 12/20/2020, 6:55:03 PM

一个BFS的基本解法, 十分的传统了。时间复杂度: $O(N)$ N 是matrix里点的个数。空间复杂度: $O(N)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
DIRECTIONS = [[1, 2], [-1, 2], [2, 1], [-2, 1]]

def shortestPath2(self, grid):
    # write your code here
    queue = deque([(0, 0)])
    path = {(0, 0): 0}
    n, m = len(grid), len(grid[0])

    while queue:
        loc = queue.popleft()
        for i, j in self.DIRECTIONS:
            x, y = loc[0] + i, loc[1] + j
            if not self.is_valid(x, y, n, m, grid):
                continue
            queue.append((x, y))
            grid[x][y] = 1
            path[(x, y)] = path[loc] + 1

    return path.get((n - 1, m - 1), -1)

def is_valid(self, x, y, n, m, grid):
    if 0 <= x < n and 0 <= y < m:
        return not grid[x][y]
    return False
```

👍 获赞 0 💬 添加评论

**X2020**

更新于 11/23/2020, 3:49:09 PM

把二维压缩为一维 $(x,y) \rightarrow (x*n)+y$

```
public class Solution { /** @param grid: a chessboard included 0 and 1 @return: the shortest path */
```

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
private int[][] dirs={{1,2}, {-1,2}, {2,1}, {-2,1}};
public int shortestPath2(boolean[][] grid) {
    int n = grid.length;
    int m = grid[0].length;
    if (n==0 || m==0) {
        return -1;
    }
    Map<Integer,Integer> distance = new HashMap<>();
    Queue<Integer> queue = new LinkedList<>();
    distance.put(0,0);
    queue.offer(0);

    while(!queue.isEmpty()) {
        int now = queue.poll();
        int now_x = now/n;
        int now_y = now%n;
        for(int[] dir:dirs) {
            int x = now_x+dir[0];
            int y = now_y+dir[1];
            if (!isValid(x, y, grid, distance)) {
                continue;
            }
            distance.put(x*n+y,distance.get(now)+1);
            queue.offer(x*n+y);
        }
    }
    int key = (n-1)*n+m-1;
    return distance.containsKey(key)?distance.get(key):-1 ;
}

public boolean isValid(int x, int y, boolean[][]grid, Map<Integer,Integer> distance) {
    int n = grid.length;
    int m = grid[0].length;
    if (x<0 || x>=n || y<0 || y>=m) {
        return false;
    }
    if (grid[x][y] || distance.containsKey(n*x+y)) {
        return false;
    }
    return true;
}
}

```

👍 获赞 0 💬 添加评论



HUE

更新于 8/15/2020, 8:33:43 AM

動態規劃版本代碼, dp 數組存 [0, 0] 到 [i, j] 的最短距離, 注意初始化 dp 數組時值要設為 INT_MAX, for 循環檢測上一步的 dp 值時要確認是否 overflow。

題目規定只能向右走, 代表列才有方向性, 行的方向則可上可下, 所以 j 要放外層, 代表先計算同一行往前的所有可能性。



invitation/sh:



```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    int shortestPath2(vector<vector<bool>> &grid) {
        // write your code here
        int m = grid.size();
        int n = grid[0].size();
        vector<vector<int>> dp(m, vector<int>(n, INT_MAX));
        dp[0][0] = 0;

        vector<vector<int>> dirs {{-1, -2}, {-2, -1}, {1, -2}, {2, -1}};
        for (int j = 0; j < n; j++) {
            for (int i = 0; i < m; i++) {
                if (grid[i][j]) {
                    continue;
                }

                for (auto dir : dirs) {
                    int x = i + dir[0];
                    int y = j + dir[1];
                    if (!isValid(grid, x, y)) {
                        continue;
                    }
                    /* Avoid Overflow */
                    if (dp[x][y] == INT_MAX) {
                        continue;
                    }

                    dp[i][j] = min(dp[i][j], dp[x][y] + 1);
                }
            }
        }

        if (dp[m - 1][n - 1] == INT_MAX) {
            return -1;
        }
        return dp[m - 1][n - 1];
    }

    bool isValid(vector<vector<bool>> &grid, int x, int y) {
        if (x < 0 || y < 0 || x >= grid.size() || y >= grid[0].size()) {
            return false;
        }
        return true;
    }
};

```

優化為滾動數組版本

j 放在外層, 且最大往回參考 2 步, 所以對 j 座標 mod 3

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    int shortestPath2(vector<vector<bool>> &grid) {
        // write your code here
        int m = grid.size();
        int n = grid[0].size();

        // DP 数组定义为从 0 出发到 (i, j) 所需的步数
        vector<vector<int>> dp(m, vector<int>(3, INT_MAX));
        dp[0][0] = 0;

        vector<vector<int>> dirs {{-1, -2}, {1, -2}, {-2, -1}, {2, -1}};
        for (int j = 1; j < n; j++) {
            for (int i = 0; i < m; i++) {
                dp[i][j % 3] = INT_MAX;
                if (grid[i][j]) {
                    continue;
                }
                for (auto dir : dirs) {
                    int x = i + dir[0];
                    int y = j + dir[1];
                    if (!isValid(grid, x, y)) {
                        continue;
                    }
                    if (dp[x][y % 3] == INT_MAX) {
                        continue;
                    }
                    dp[i][j % 3] = min(dp[i][j % 3], dp[x][y % 3] + 1);
                }
            }
        }

        if (dp[m - 1][(n - 1) % 3] == INT_MAX) {
            return -1;
        }
        return dp[m - 1][(n - 1) % 3];
    }

    bool isValid(vector<vector<bool>> &grid, int x, int y) {
        if (x < 0 || y < 0 || x >= grid.size() || y >= grid[0].size()) {
            return false;
        }
        return true;
    }
};

```

双向 BFS 版本

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /

```

Resume / Project 2020版

- * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
- * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
- * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code

*/

```
struct PointNode {
    int x;
    int y;
    PointNode(int a, int b) : x(a), y(b) {}
};
```

```
class Solution {
public:
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    int shortestPath2(vector<vector<bool>> &grid) {
        PointNode source(0, 0);
        PointNode destination(grid.size() - 1, grid[0].size() - 1);

        if (grid.empty() || grid[0].empty()) {
            return -1;
        }
        if (source.x == destination.x && source.y == destination.y) {
            return 0;
        }
        if (grid[source.x][source.y] || grid[destination.x][destination.y]) {
            return -1;
        }

        int m = grid.size(), n = grid[0].size();

        queue<PointNode> forwardQueue;
        unordered_set<int> forwardSet;
        queue<PointNode> backwardQueue;
        unordered_set<int> backwardSet;

        forwardQueue.push(source);
        forwardSet.insert(source.x * n + source.y);
        backwardQueue.push(destination);
        backwardSet.insert(destination.x * n + destination.y);

        vector<vector<int>> forwardDirs {{1, 2}, {-1, 2}, {-2, 1}, {2, 1}};
        vector<vector<int>> backwardDirs {{2, -1}, {1, -2}, {-2, -1}, {-1, -2}};
        int distance = 0;
        while (!forwardQueue.empty() && !backwardQueue.empty()) {
            distance++;
            if (extendQueue(grid, forwardQueue, forwardSet, backwardSet, forwardDirs)) {
                return distance;
            }

            distance++;
            if (extendQueue(grid, backwardQueue, backwardSet, forwardSet, backwardDirs)) {
                return distance;
            }
        }

        return -1;
    }

    bool extendQueue(vector<vector<bool>> &grid,
                    queue<PointNode> &q,
                    unordered_set<int> &visited,
                    unordered_set<int> &oppositeVisited,
                    vector<vector<int>> &dirs) {
        int m = grid.size(), n = grid[0].size();
```

```

    int size = q.size();
    for (int i = 0; i < size; i++) {
        int x = q.front().x;
        int y = q.front().y;
        q.pop();
        for (auto dir : dirs) {
            int nx = x + dir[0];
            int ny = y + dir[1];
            if (!isValid(grid, visited, nx, ny)) {
                continue;
            }
            if (oppositeVisited.count(nx * n + ny)) {
                return true;
            }
            q.push(PointNode(nx, ny));
            visited.insert(nx * n + ny);
        }
    }
    return false;
}

bool isValid(vector<vector<bool>> &grid,
            unordered_set<int> &visited,
            int x,
            int y) {
    int m = grid.size(), n = grid[0].size();
    if (x < 0 || x >= grid.size() || y < 0 || y >= grid[0].size()) {
        return false;
    }
    if (grid[x][y]) {
        return false;
    }
    if (visited.count(x * n + y)) {
        return false;
    }
    return true;
}
};

```

👍 获赞 0 💬 添加评论



Peter

更新于 6/23/2020, 3:12:45 AM

打印骑士的最短路径

- Used a HashMap to remember a mapping between:
(newX, newY) -> (x, y)

```

/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
}

```

```
public int shortestPath2(boolean[][] grid) {
    if (grid == null || grid.length == 0 || grid[0].length == 0) {
        return -1;
    }
    int m = grid.length;
    int n = grid[0].length;
    boolean[][] visited = new boolean[m][n];

    Queue<int[]> q = new LinkedList<>();
    q.offer(new int[]{0, 0});
    visited[0][0] = true;
    Map<int[], int[]> lastPoint = new HashMap<>();

    int[] dirX = {1, -1, 2, -2};
    int[] dirY = {2, 2, 1, 1};
    int step = 0;

    while(!q.isEmpty()) {
        int qSize = q.size();
        for (int index = 0; index < qSize; index++) {
            int[] temp = q.poll();
            int x = temp[0];
            int y = temp[1];
            if (x == m - 1 && y == n - 1) {
                List<int[]> path = findPath(grid, lastPoint);
                int pathSize = path.size();
                int count = 0;
                System.out.println("This is the shortest path: ");
                for(int[] p: path) {
                    System.out.print(Arrays.toString(p));
                    if (count != pathSize - 1) {
                        System.out.print("->");
                    }
                    count++;
                }
                return step;
            }
            for (int i = 0; i < 4; i++) {
                int newX = x + dirX[i];
                int newY = y + dirY[i];
                if (newX < 0 || newX >= m || newY < 0 || newY >= n) {
                    continue;
                }
                if (visited[newX][newY] || grid[newX][newY] == true) {
                    continue;
                }
                q.offer(new int[]{newX, newY});
                visited[newX][newY] = true;
                lastPoint.put(new int[]{newX, newY}, new int[]{x, y});
            }
            step++;
        }
    }

    return -1;
}

private List<int[]> findPath(boolean[][] grid, Map<int[], int[]> lastPointMap) {
    int m = grid.length;
    int n = grid[0].length;

    int x = m - 1;
    int y = n - 1;
    List<int[]> path = new ArrayList<>();
    int[] point = new int[]{x, y};
    for (Map.Entry<int[], int[]> entry: lastPointMap.entrySet()) {
        if (entry.getKey()[0] == m - 1 && entry.getKey()[1] == n - 1) {
            point = entry.getKey();
        }
    }
    path.add(point);
    while (point != null) {
        point = lastPointMap.get(point);
    }
    return path;
}
```



```

    }
}
path.add(new int[]{point[0], point[1]});
while (true) {
    if (x == 0 && y == 0) {
        break;
    }
    int[] prevPoint = lastPointMap.get(point);
    x = prevPoint[0];
    y = prevPoint[1];
    for (Map.Entry<int[], int[]> entry: lastPointMap.entrySet()) {
        if (entry.getKey()[0] == x && entry.getKey()[1] == y) {
            point = entry.getKey();
        }
    }
    path.add(new int[]{x, y});
}

Collections.reverse(path);
return path;
}
}

```

👍 获赞 0 💬 添加评论



九章用户BHIJVP5

更新于 6/9/2020, 7:04:24 AM

同骑士原题, 而且可以走的方向只有四个。起点是 0 (); 重点是 row - 1 ();

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Point {
    int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

public class Solution {
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    public int shortestPath2(boolean[][] grid) {
        // write your code here
        if (grid == null || grid.length == 0 || grid[0].length == 0) {
            return -1;
        }

        int row = grid.length;
        int col = grid[0].length;

        if (row == 1 && col == 1) {
            return 0;
        }
    }
}

```

```
Queue<Point> queue = new LinkedList<>();
queue.offer(new Point(0, 0));
grid[0][0] = true;

int[] deltaX = {1, -1, 2, -2};
int[] deltaY = {2, 2, 1, 1};
int result = 0;

while (!queue.isEmpty()) {
    int size = queue.size();
    result++;
    for (int i = 0; i < size; i++) {
        Point current = queue.poll();
        for (int j = 0; j < 4; j++) {
            Point next = new Point(current.x + deltaX[j], current.y + deltaY[j]);
            if (!inBound(grid, next) || grid[next.x][next.y]) {
                continue;
            }

            if (next.x == row - 1 && next.y == col - 1) {
                return result;
            }

            queue.offer(next);
            grid[next.x][next.y] = true;
        }
    }
}

return -1;
}

private boolean inBound(boolean[][] grid, Point next) {
    int row = grid.length;
    int col = grid[0].length;

    return next.x >= 0 && next.x < row && next.y >= 0 && next.y < col;
}
}
```

👍 获赞 0 💬 添加评论



Chebyshev

更新于 6/9/2020, 7:04:22 AM

分层遍历的广度优先搜索。访问过的点用1来标记, 防止重复访问造成TLE。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    int shortestPath2(vector<vector<bool>> &grid) {
        // write your code here
        if (grid.empty()) {
            return -1;
        }

        int m = grid.size(), n = grid[0].size();
        queue<vector<int>> q; q.push({0, 0});

        // define direction vectors
        vector<int> dx{1, -1, 2, -2}, dy{2, 2, 1, 1};
        int count = 0;

        while (!q.empty()) {
            int size = q.size();
            for (int i = 0; i < size; ++i) {
                vector<int> node = q.front(); q.pop();

                // check if there is obstacle or out of boundary.
                if (!check(grid, node[0], node[1])) {
                    continue;
                }

                // check if we arrive the terminal.
                if (node[0] == (m - 1) && node[1] == (n - 1)) {
                    return count;
                }

                grid[node[0]][node[1]] = 1;
                for (int j = 0; j < 4; ++j) {
                    q.push({node[0] + dx[j], node[1] + dy[j]});
                }
            }
            ++count;
        }

        return -1;
    }

    bool check(vector<vector<bool>> &grid, int x, int y) {
        int m = grid.size(), n = grid[0].size();

        if (0 <= x && x < m && 0 <= y && y < n && grid[x][y] == 0) {
            return true;
        }

        return false;
    }
};

```

👍 获赞 0 💬 添加评论

**kevin**

更新于 6/9/2020, 7:04:18 AM

这道题用bfs应该更容易理解一些, 但为了练习dp, 写了这个dp的方法。一个tricky的地方, 要从列开始循环。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param grid: a chessboard included 0 and 1
    @return: the shortest path
    """
    def shortestPath2(self, grid):
        # write your code here
        dirs = [[1, 2], [-1, 2], [2, 1], [-2, 1]]
        m = len(grid)
        n = len(grid[0])

        dp = [[sys.maxsize] * n for _ in range(m)]
        dp[0][0] = 0

        for y in range(n): #because only jump to right, so scan column
            for x in range(m):
                if grid[x][y] == 1:
                    continue

                for dx, dy in dirs:
                    pre_x = x - dx
                    pre_y = y - dy
                    if pre_x < 0 or pre_x >= m or pre_y < 0 or pre_y >= n or dp[pre_x][pre_y] == sys.maxsize:
                        continue

                    dp[x][y] = min(dp[x][y], dp[pre_x][pre_y] + 1)

        if dp[m - 1][n - 1] == sys.maxsize:
            return -1
        else:
            return dp[m - 1][n - 1]
```

👍 获赞 0

💬 添加评论

**EricC**

更新于 6/9/2020, 7:04:15 AM

比较好想到的2维DP 超详细的思路 and 注解, 请参考code里的comments。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
```

```

*/
/**
 * DP (2 - D)      T: O (n ^ 2)
 *
 * Idea: maintain a state of shortest path from 0,0 -> i,j.
 * state: dp[i][j]: shortest path from 0,0 -> i,j
 * state transition:
 * conditions:
 * only if
 * 1. grid[i][j] is not barrier (grid[i][j])
 * 2. dp[i][j] unvisited (dp[i][j] == -1)
 * 3. dp[i - deltaX][j - deltaY] exists (dp[i - deltaX][j - deltaY] != -1)
 * then
 * dp[i][j] = min(dp[i - deltaX][j - deltaY]) + 1;
 * state initialization:
 * dp[0][0] = 0, else dp[i][j] = -1.
 *
 * Notice:
 * 1. since knight moves left to right only, therefore 1 grid can not be visited once, ignore when visited.
 * 2. tracking from left to right, not top to bottom.
 * 3. since there might be multiple ways to get to same grid, need to calculate backwards, say
 *     dp[i][j] from dp[i - deltaX][j - deltaY]
 *     not
 *     dp[i][j] to dp[i + deltaX][j + deltaY].
 */
public class Solution {
    /**
     * @param grid: a chessboard included 0 and 1
     * @return: the shortest path
     */
    public int shortestPath2(boolean[][] grid) {
        if (grid == null || grid.length == 0 || grid.length < 2 && grid[0].length < 2) {
            return -1;
        }

        int[][] dp = new int[grid.length][grid[0].length];
        int[] deltaX = new int[]{1, -1, 2, -2};
        int[] deltaY = new int[]{2, 2, 1, 1};


        // init
        for (int i = 0; i < dp.length; i++) {
            for (int j = 0; j < dp[0].length; j++) {
                if (i == 0 && j == 0) {
                    dp[i][j] = 0;
                    continue;
                }
                dp[i][j] = -1;
            }
        }

        // tracking min of previous path. from left to right, not top to bottom
        for (int j = 0; j < dp[0].length; j++) {
            for (int i = 0; i < dp.length; i++) {
                if (dp[i][j] != -1 || grid[i][j]) {
                    continue;
                }
                int min = Integer.MAX_VALUE;
                for (int k = 0; k < deltaX.length; k++) {
                    if (i - deltaX[k] >= 0 && i - deltaX[k] < dp.length && j - deltaY[k] >= 0 && dp[i - deltaX[k]][j - deltaY[k]] != -1) {
                        min = Math.min(min, dp[i - deltaX[k]][j - deltaY[k]]);
                    }
                }
                dp[i][j] = min == Integer.MAX_VALUE ? -1 : min + 1;
            }
        }

        return dp[grid.length - 1][grid[0].length - 1];
    }
}

```

```
}  
}
```

 获赞 0  添加评论



九章用户QXK7A3
更新于 6/9/2020, 7:04:08 AM

双向bfs的算法，从java的算法转过来的，注意要单独拿出来的特殊情况相较于一般的bfs要多一些。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param grid: a chessboard included 0 and 1
    @return: the shortest path
    """
    def shortestPath2(self, grid):
        # write your code here
        if not grid or len(grid[0]) == 0 or grid[0][0] == 1:
            return -1
        if grid[-1][-1]:
            return -1
        r, c = len(grid), len(grid[0])
        if r == c == 1:
            return 0

        pos = [(1, 2), (-1, 2), (2, 1), (-2, 1)]

        usedA = {0:True}
        que_A = collections.deque([0])

        usedB = {r*c-1:True}
        que_B = collections.deque([r*c-1])

        curr_que = None
        curr_set = None
        oppo = None
        res = 0

        while que_A and que_B:
            if len(que_A) > len(que_B):
                curr_que = que_B
                curr_set = usedB
                sign = -1
                oppo = usedA
            else:
                curr_que = que_A
                curr_set = usedA
                sign = 1
                oppo = usedB
            L = len(curr_que)
            res += 1
            for i in range(L):
                idx = curr_que.popleft()
                x, y = idx//c, idx%c
                for dx, dy in pos:
                    new_x = x+sign*dx
                    new_y = y+sign*dy
                    if 0 <= new_x < r and 0 <= new_y < c and grid[new_x][new_y] != 1:
                        if new_x*c + new_y in oppo:
                            return res
                        if new_x*c + new_y not in curr_set:
                            curr_set[new_x*c+new_y] = True
                            curr_que.append(new_x*c+new_y)
            return -1

```

👍 获赞 0

💬 添加评论

**steinsgate**

更新于 6/9/2020, 7:04:00 AM

这个同样是滚动数组优化和令狐冲老师的滚动数组的解法的区别在于不需要每次进来都额外花 $O(3n)$ 的时间取重置现在要填的列和接下去2列的f值。通过每次到fi ()这个点只要不是原点就可以重置。因为转移方程计算fi ()并不依赖当前位置的fi ()值。所以可以在处理到fi ()这个点的时候先重置再做后续计算。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
DELTA DIRECTIONS = [
    (1, 2),
    (-1, 2),
    (2, 1),
    (-2, 1)
]

class Solution:
    """
    @param grid: a chessboard included 0 and 1
    @return: the shortest path
    """
    def shortestPath2(self, grid):
        # write your code here
        if not grid or not grid[0]:
            return -1
        n = len(grid)
        m = len(grid[0])
        f = [[sys.maxsize] * 3 for _ in range(n)]

        f[0][0] = 0 #初始状态
        for j in range(0, m):
            for i in range(0, n):
                if i == 0 and j == 0: #初始状态得绕过不然会覆盖成sys.maxsize。除了原点以外的都是可以遍历的
                    continue
                if grid[i][j]:
                    f[i][j % 3] = sys.maxsize #当是石头的时候也得标记程sys.maxsize, 不然会留下上次滚动数组的值, 影响右边的计算
                    continue
                f[i][j % 3] = sys.maxsize
                for delta_x, delta_y in DELTA DIRECTIONS:
                    new_x, new_y = i - delta_x, j - delta_y
                    if self.is_valid(grid, n, m, new_x, new_y):
                        f[i][j % 3] = min(f[i][j % 3], f[new_x][new_y % 3] + 1)
        return f[n - 1][(m - 1) % 3] if f[n - 1][(m - 1) % 3] != sys.maxsize else -1

    def is_valid(self, grid, n, m, x, y):
        return 0 <= x < n and 0 <= y < m
```

👍 获赞 0

💬 添加评论

进阶课程

视频+互动	直播+互动	直播+互动	互动课
<div>九章算法班 2021 版</div> <div>8周时间精通 57 个核心高频考点，9 招击破 FLAG、BATJ 算法面试。22....</div>	<div>系统架构设计 System Design 2021 版</div> <div>成为百万架构师必上。30 课时带你快速掌握 18大系统架构设计知识点与面...</div>	<div>九章算法面试高频题冲刺班</div> <div>每期更新 15% 题目，考前押题，一举拿下FLAG & BATJ Offer</div>	<div>面向对象设计 OOD</div> <div>应届生及亚马逊面试必考，IT求职必备基础</div>