

1. Least Substring

<https://www.lintcode.com/problem/least-substring/description>

代码：

```
public class Solution {  
    /**  
     * @param s: the string s  
     * @param k: the maximum length of substring  
     * @return: return the least number of substring  
     */  
    public int getAns(String s, int k) {  
        // Write your code here  
        int n = s.length(), count = 1, ans = 1;  
        for(int i = 1; i < n; i++) {  
            if(count < k && s.charAt(i) == s.charAt(i - 1)) {  
                count++;  
            } else {  
                ans++;  
                count = 1;  
            }  
        }  
        return ans;  
    }  
}
```

1. K-Substring with exactly K different characters(长度为 K , 有 K 个不同)

<https://www.lintcode.com/problem/k-substring-with-k-different-characters/description?from=ladder&&fromId=69>

<https://www.jiuzhang.com/solution/k-substring-with-k-different-characters/>

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=485608&extra=page%3D3%26filter%3Dauthor%26orderby%3Ddateline%26sortid%3D192%26sortid%3D192%26orderby%3Ddateline>

注意审题，可能有变种

双指针，O(n)复杂度

```
public int KSubstring(String stringIn, int K) {  
    if (stringIn == null || stringIn.length() == 0 || K <= 0) {  
        return 0;  
    }  
    int count = 0;  
    HashMap<Character, Integer> charMap = new HashMap<>();  
    HashSet<String> resultSet = new HashSet<String>();  
    int len = stringIn.length();
```

```

int j = 0;
for (int i = 0; i < len; i++) {
    while (j < len && charMap.size() < K) {
        char c = stringIn.charAt(j);
        if (charMap.containsKey(c)) {
            break;
        }
        charMap.put(c, 1);
        j++;
        if (charMap.size() == K) {
            resultSet.add(stringIn.substring(i, j));
        }
    }
    charMap.remove(stringIn.charAt(i));
}
return resultSet.size();
}

```

1. TODO: Return all the substring with K characters contain K-1 distinct characters

(长度为 K, 有 K - 1 个不同)

<https://www.1point3acres.com/bbs/thread-483122-1-1.html>

<https://leetcode.com/discuss/interview-question/124652/Amazon-onsite-interview-question>

<https://github.com/ericster/workspaceMars/blob/0bb98a042f029d4a83ad35213c313973a7263c32/subStringsLessKDist/src/subStringsLessKDist/Solution.java>

<https://ideone.com/9ToAFo>

Michelle has created a word game for her students. The word game begins with Michelle writing a string and a number, K, on the board.

The students must find a substring of size K such that there is exactly one character that is repeated one;

in other words, there should be k - 1 distinct characters in the substring.

Write an algorithm to help the students find the correct answer. If no such substring can be found, return an empty list;

if multiple such substrings exist, return all them, without repetitions. The order in which the substrings are does not matter.

Input:

The input to the function/method consists of two arguments - inputString, representing the string written by the teacher;

num an integer representing the number, K, written by the teacher on the board.

Output:

Return a list of all substrings of inputString with K characters, that have k-1 distinct character i.e. exactly one character is repeated, or an empty list if no such substring exist in inputString.

The order in which the substrings are returned does not matter.

Constraints:

The input integer can only be greater than or equal to 0 and less than or equal to 26 (0 <= num <= 26)

The input string consists of only lowercase alphabetic characters.

Example

Input:

inputString = awaglk

num = 4

Output:

[awag]

Explanation:

The substrings are {awag, wagl, aglk}

The answer is awag as it has 3 distinct characters in a string of size 4, and only one character is repeated twice.

此题有 16 个隐藏 Testcase, 其中 8 个的情况应该是重复的 Substring, 用 Set 保存结果然后存到 List 里面, 最后通过 16 个 case。

With Michelle writing a string and a number, K, on the board. The students must find a substring of size K such that there is exactly one character that is repeated once; in other words, there should be K – 1 distinct characters in the substring.
Write an algorithm to help the students find the correct answer. If no such substring can be found, return an empty list; if multiple such substrings exist, return all of them, without repetitions. The order in which the substrings are returned does not matter.
Input
The input to the function/method consists of two arguments - *inputString*, representing the string written by the teacher; *num*, an integer representing the number, K, written by the teacher on the board.
Output
Return a list of all substrings of *inputString* with K characters, that have K-1 distinct character i.e. exactly one character is repeated, or an empty list if no such substring exists in *inputString*. The order in which the substrings are returned does not matter.
Constraints
The input integer can only be greater than or equal to 0 and less than or equal to 26 ($0 \leq num \leq 26$)
The input string consists of only lowercase alphabetic characters.
Examples

```
1 // IMPORT LIBRARY PACKAGES NEEDED BY YOUR PROGRAM
2 // SOME CLASSES WITHIN A PACKAGE MAY BE RESTRICTED
3 import java.util.List;
4 // DEFINE ANY CLASS AND METHOD NEEDED
5 // CLASS BEGINS, THIS CLASS IS REQUIRED
6 public class Solution
7 {
8     // METHOD SIGNATURE BEGINS, THIS METHOD IS REQUIRED
9     public List<String> subStringsLessKDist(String inputString, int num)
10    {
11        // WRITE YOUR CODE HERE
12    }
13    // METHOD SIGNATURE ENDS
14 }
```

1. **Return the number of all the substring with K distinct characters(没有限制子串长度, 只要有 K 个 distinct 就行, 而且不需要限制子串必须 distinct, 因此不用 set())**

Willy Wonka devised a fun game for the city to help identify...

```
int countKDistinctSubstrings(String inputString, int num) {
    if(inputString == null || inputString.length() == 0 || num == 0 || num > inputString.length()) {
        return 0;
    }

    int res = 0;
    int n = inputString.length();
    int[] count = new int[26];

    for(int i = 0; i < n; i++) {
        int distinctNum = 0;
        Arrays.fill(count, 0);
        for(int j = i; j < n; j++) {
            count[inputString.charAt(j) - 'a']++;
            if(count[inputString.charAt(j) - 'a'] == 1) {
                distinctNum++;
            } else if(count[inputString.charAt(j) - 'a'] > 1) {
                distinctNum--;
            }
            if(distinctNum == num) {
                res++;
            }
        }
    }
    return res;
}
```

```

        for(int j = i; j < n; j++) {
            char c = inputString.charAt(j);
            if(count[c - 'a'] == 0) {
                distinctNum++;
            }
            count[c - 'a']++;
            if(distinctNum == num) {
                res++;
            }
        }
    }
    return res;
}

```

2. Number of restaurants

<https://www.lintcode.com/problem/number-of-restaurants/description?from=ladder&&fromId=69>

```

public class Solution {
    /**
     * @param restaurant:
     * @param n:
     * @return: nothing
     */
    public List<List<Integer>> nearestRestaurant(List<List<Integer>> restaurant, int n) {
        // Write your code here
        if(restaurant == null || restaurant.size() == 0 || n > restaurant.size()) {
            return null;
        }

        List<List<Integer>> ans = new ArrayList<>();
        int furthestDistance = 0;
        int[] distances = new int[restaurant.size()];

        for(int i = 0; i < distances.length; i++) {
            distances[i] = calculateDistance(restaurant.get(i));
        }

        Arrays.sort(distances);
        furthestDistance = distances[n - 1];

        for(int i = 0; i < restaurant.size() && ans.size() < n; i++) {
            if(calculateDistance(restaurant.get(i)) <= furthestDistance) {
                ans.add(restaurant.get(i));
            }
        }
    }
}

```

```

        return ans;
    }

private int calculateDistance(List<Integer> point) {
    return point.get(0) * point.get(0) + point.get(1) * point.get(1);
}
}

```

3. Aerial Movie 飞机上的电影

https://www.lintcode.com/problem/aerial-movie/description?_from=ladder&&fromId=69

```

public class Solution {
    /**
     * @param t: the length of the flight
     * @param dur: the length of movies
     * @return: output the lengths of two movies
     */
    public int[] aerial_Movie(int t, int[] dur) {
        // Write your code here
        // double pointers
        if(dur == null || dur.length == 0 || t <= 30) {
            return new int[]{};
        }
        Arrays.sort(dur);
        t -= 30;
        int[] ans = new int[2];
        int low = 0, high = dur.length - 1;
        int diff = Integer.MAX_VALUE;
        while(low < high) {
            int total = dur[low] + dur[high];
            if(total == t) {
                return new int[]{dur[low], dur[high]};
            } else if (total < t) {
                if(t - total < diff) {
                    diff = t - total;
                    ans[0] = dur[low];
                    ans[1] = dur[high];
                }
                low++;
            } else {
                high--;
            }
        }
        return ans;
    }
}

```

4. **getMostPopularNode** (general tree, leaves not counted), return tree node, Maximum Average Sum Subtree

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=478916>

The screenshot shows a programming challenge interface with the following details:

- Time Left:** 06m 36s
- QUESTION 2 out of 2**
- aspirinGminds**
- HELP 0**
- Problem | Test Cases | Output |**
- Description:** The current selected programming language is CPP. We emphasize the submission of a fully working code over partially correct but efficient code. Once submitted, you cannot review this problem again. You can use `cout` to debug your code. The `cout` may not work in case of syntax/runtime error. The version of OCC being used is 8.1.0.
- Problem Statement:** A product category tree represents the hierarchy of product categories sold by Amazon. Product categories have child categories within them, and these child categories can, in turn, have child categories within them. Each node in the tree contains an integer that represents the number of visits to that category's page. A customer can go directly to a child category without going to the parent category. Category popularity is computed as the average visits to all child categories and the category itself. The most popular parent category has the highest category popularity. A parent category must have child categories to be a parent category.
- Input:** Write an algorithm to find the node which is the most popular parent category.
- Input:** The input to the function/method consists of an argument - `rootCategory`, representing the root node of the product category tree.
- Output:** Return the node which is the most popular parent category.
- Note:** There will be at least one child category in the tree and that there will be no ties.
- Example:**

```
Input:
rootCategory = 
  20
  / \
  12  18
 / \ / \
11 3 15 8

Output:
18
```
- Explanation:** There are three parent categories in this tree with the following category popularity:
 $12 \Rightarrow (11+2+3+12)/4 = 7$
 $18 \Rightarrow (18+15+8)/3 = 13.67$
 $20 \Rightarrow (12+11+2+3+18+15+8+20)/8 = 11.125$
The most popular parent category is the parent category with 18 visits.
So, the output is 18.
- Helper Description:** The following class is used to represent a `CategoryNode` and is already implemented in the default code (Do not write this definition again in your code):

```
class CategoryNode
{
public:
    int value;
    vector<CategoryNode> subCategoryNode;
    CategoryNode();
    CategoryNode(int tenure);
    {
        this->value = tenure;
    }
};
```

Problem | **Test Cases** | **Output** |

The current selected programming language is **Java**. We emphasize the submission of a fully working code over partially correct but efficient code. Once **submitted**, you cannot review this problem again. You can use `System.out.println()` to debug your code. The `System.out.println()` may not work in case of syntax/runtime error. The version of **JDK** being used is **1.8**.

A product category tree represents the hierarchy of product categories sold by Amazon. Product categories have child categories within them, and these child categories can, in turn, have child categories within them. Each node in the tree contains an integer that represents the number of visits to that category's page. A customer can go directly to a child category without going to the parent category.

Category popularity is computed as the average visits to all child categories and the category itself. The most popular parent category has the highest category popularity. A parent category must have child categories to be a parent category.

Write an algorithm to find the node which is the most popular parent category.

Input

The input to the function/method consists of an argument - `rootCategory`, representing the root node of the product category tree.

Output

Return the node which is the most popular parent category.

Note

There will be at least one child category in the tree and that there will be no ties.

Example

Input:

```
rootCategory =  
20
```

Compile and Run



```
1 // IMPORT LIBRARY PACKAGES NEEDED BY YOUR PROGRAM  
2 // SOME CLASSES WITHIN A PACKAGE MAY BE RESTRICTED  
3 // DEFINE ANY CLASS AND METHOD NEEDED  
4 // EmployeeNode CLASS IS ALREADY DEFINED  
5 // CLASS BEGINS, THIS CLASS IS REQUIRED  
6 public class Solution  
7 {  
8     // METHOD SIGNATURE BEGINS, THIS METHOD IS REQUIRED  
9     public CategoryNode getMostPopularNode(CategoryNode rootCategory)  
10    {  
11        // WRITE YOUR CODE HERE  
12    }  
13    // METHOD SIGNATURE ENDS  
14 }
```

```

// CLASS BEGINS, THIS CLASS IS REQUIRED
import java.util.*;

public class Solution
{
    class Result {
        CategoryNode node;
        int sum;
        int size;
        public Result(CategoryNode node, int sum, int size){
            this.node = node;
            this.sum = sum;
            this.size = size;
        }
    }

    private Result res = null;

    // METHOD SIGNATURE BEGINS, THIS METHOD IS REQUIRED
    public CategoryNode getMostPopularNode(CategoryNode rootCategory)
    {
        if (rootCategory == null){
            return null;
        }
        Result rootRes = helper(rootCategory);
        return res.node;
    }
    // METHOD SIGNATURE ENDS

    public Result helper(CategoryNode root){
        if (root == null){
            return new Result(null, 0, 0);
        }

        ArrayList<Result> results = new ArrayList<Result>();
        for(CategoryNode n: root.subCategoryNode){
            results.add(helper(n));
        }
        int childSum = 0;
        int childSize = 0;
        for (Result r: results){
            childSum += r.sum;
            childSize += r.size;
        }
        Result curRes = new Result(root, childSum + root.value, childSize + 1);
        if (res == null || curRes.sum * res.size > res.sum * curRes.size){
            if (curRes.size > 1)
                res = curRes;
        }
        return curRes;
    }

}

```

Index: getMostPopularNode

5. **getFastestComponent()**

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=480213>

The screenshot shows a programming challenge interface. At the top, there are tabs for 'Problem', 'Test Cases', and 'Output'. On the right, there are icons for copy, paste, and other functions, with 'Java' selected. A progress bar indicates '05m 10s' has passed. To the right, there are two green circles labeled '1' and '2'. The main area contains the following text:

The current selected programming language is Java. We emphasize the submission of a fully working code over partially correct but efficient code. Once submitted, you cannot review this problem again. You can use `System.out.println()` to debug your code. The version of JDK being used is 1.8.

Engineers at Amazon are interested in knowing which software component moves the fastest by counting the number of lines of code changed in that component and all subcomponents in the code base over a month. Component nodes have subcomponent nodes that make up the larger component. We've stored the components as a tree where each node represents one component that contains the number of lines changed that month. Software component speed is computed as the average number of lines of code changed for a given component and all its subcomponents. Engineers compute this software component speed for all such components which have one or more subcomponents. Find the component with the highest software component speed. Assume there will be at least one subcomponent in the tree and that there will be no ties.

Input:
The input to the function/method consists of an argument:
`rootComponent`, representing the root node.

Output:
Return the reference to the component that has highest software component speed.

Example:
Input:
`rootComponent:`
200
/ \
120 180
/ | \ / \\
110 20 30 150 80

Output:
180

Explanation:
The component at the root of the tree has had 200 lines changed this month; its two dependencies have had 120 and 180 changed respectively.
There are three components in this tree with the following changes:
 $120 \Rightarrow (110+20+30+120)/4 = 70$
 $180 \Rightarrow (180+150+80)/3 = 130.67$

6. Two sum Cloest 货车装货问题

1. two sum closest(\leq target)

Input:

the input to the function/method consists of three arguments, numContainers, an integer representing the number of containers; maxCapacity, a real number representing the maximum capacity of the shelf; containerWt, a list of real numbers representing the weight of the containers

Output:

Return a pair of real numbers representing the weights of the two heaviest containers chosen to be carried on the shelf.

Example:

Input:

```
num Containers = 6
maxCapacity=43.3
containersWt=[7.33,8.13,103.73,13,11.24,23.79,18.35]
```

Output:

[23.79, 18.35]

7. Highest Five

<https://yeqiuquan.blogspot.com/2017/03/lintcode-613-high-five.html>

Problem | **Test Cases** | **Output** |

At Amazon, we focus on the customer, and one important metric is how long a customer waits for a page to render. Every time we render a page we record the page rendered, date and time the page was rendered, and how long it took to render the page. To help assess the customer experience we want to know how long on average it is taking to render each page, but we are looking for outliers. To calculate these outliers we take the five longest (worst) rendering times for a page and average them together. Assume that each page has been rendered at least five times.

Write an algorithm to find the average of five worst rendering times for each page.
Input

The input to the function/method consists of two arguments-
renderCount, an integer representing the total number of pages to be rendered (includes the repeated rendering of the same page),
renderTimeOfPages, a list where each element of the list consists of an integer representing the page ID and a real number representing the rendering time of the page.

Output

Return a list where each element of the list consists of an integer representing the page ID and a real number representing the average of five worst rendering times of each page.

Example

Input:

```
renderCount = 13
renderTimeOfPages =
[[pageID: 1, renderTime: 40],
 [pageID: 2, renderTime: 90],
 [pageID: 1, renderTime: 50].
```

The screenshot shows a Java code editor interface. On the left, there's a sidebar with tabs for 'Problem', 'Test Cases', 'Output', and 'Constraints'. The 'Problem' tab is active, displaying the following text:

Customers trust Amazon's reviews to make their buying decisions. The customers can see the reviews in different ways. One way to look at the reviews is by seeing the extremes, such as the most positive aggregate scores. The most positive aggregate score is the average of the top 5 highest review scores.

Given a list of product ID's and review scores, write an algorithm to find the most positive aggregate for each product.

Input
The input to the function/method consists of two arguments -
scoreCount, an integer representing the count of review scores for all the products.
reviewScoresOfProduct, a list of numbers where each element of the list consists of an integer representing the product ID and a real number representing the review score of the product.

Output
Return a list of numbers where each element of the list consists of an integer representing the product ID and a real number representing the most positive aggregate score of the product.

Constraints
 $5 \leq scoreCount$

On the right, the code editor displays a template class named 'HighestFive' with a method 'calculateHighestFive'. The code uses imports for ArrayList, Map, Comparator, HashMap, and PriorityQueue. It defines a class 'HighestFive' with a method 'calculateHighestFive' that takes 'scoreCount' and 'reviewScoresOfProduct' as parameters. The implementation starts with a comment '// WRITE YOUR CODE HERE' and includes logic to handle the input and calculate the result.

SUBMIT TEST SUBMIT ANSWER

2. Highest Five Input: 一个二维数组（实际是给出的[ProductReviewScore] class），[[product id, val], [product id, val], ...]

要求找到每个product id 对应的val中最大5个数的平均值。

输出：[[product id, average value of top 5], [product id, average value of top 5], ...]

eg:

假设输入为 [[id=1, 1], [id = 2, 2], [id=1, 3], [id = 2, 4], [id=1, 5], [id = 2, 6], [id=1, 7], [id = 2, 8], [id=1, 9], [id = 2, 10], [id=1, 11], [id = 2, 12]]

id=1, 对应的 val 有 1,3,5,7,9,11. 故得到最大5个数的平均值 $(3+5+7+9+11) / 5$

id=2, 对应的 val 有 2,4,6,8,10,12,故得到最大5个数的平均值 $(4+6+8+10+12) / 5$

注意给出的product review score的type也是double。

8. K closest points

<https://www.1point3acres.com/bbs/thread-457483-1-1.html>

<https://www.1point3acres.com/bbs/thread-469583-1-1.html>

1. Amazon Fresh 题目

Problem | Test Cases | Output |

QUESTION
1 out of 2

89m
42s

use `System.out.println()` to debug your code. The `System.out.println()` may not work in case of syntax/runtime error. This version of JDK being used is 1.8.

Amazon Fresh is a grocery delivery service that offers consumers the option of purchasing their groceries online and schedule future deliveries of purchased groceries. Amazon's backend system dynamically tracks each Amazon Fresh delivery truck and automatically assigns the next deliveries in a truck's plan. To accomplish this, the system generates an optimized delivery plan with X destinations. The most optimized plan would deliver to the closest X destinations from the start among all of the possible destinations in the plan.

Input

The input to the function/method consists of three arguments:

- numDestinations*, an integer representing the total number of possible delivery destinations for the truck (N);
- allLocations*, a list where each element consists of a pair of integers representing the x and y coordinates of the delivery locations;
- numDeliveries*, an integer representing the number of deliveries that will be delivered in the plan (X).

Output

Return a list of elements where each element of the list represents the x and y integer coordinates of the delivery destinations.

Constraints

numDeliveries < numDestinations

Note

The plan starts from the truck's location [0, 0]. The distance of the truck from a delivery destination (x, y) is the square root of $x^2 + y^2$. If there are ties then return any of the locations as long as you satisfy returning X deliveries.

Example

Input:
numDestinations = 3
allLocations = [[0, 2], [3, 4], [1, -1]]
numDeliveries = 2

Output:
[[1, -1], [0, 2]]

Explanation:
The distance of the truck from location [1, 2] is square root(5) = 2.236
The distance of the truck from location [3, 4] is square root(25) = 5
The distance of the truck from location [1, -1] is square root(2) = 1.414
The distance of the truck from location [0, 2] is square root(4) = 2
numDeliveries is 2, hence the output is [1, -1] and [0, 2].

Compile & Run

```

1 // IMPORT LIBRARY PACKAGES NEEDED BY YOUR PROGRAM
2 // SOME CLASSES WITHIN A PACKAGE MAY BE RESTRICTED
3 // TO THE JAVAC AND METHOD NEEDED
4 import java.util.List;
5 // CLASS BEGINS; THIS CLASS IS REQUIRED
6 public class Solution
7 {
8     // METHOD SIGNATURE BEGINS, THIS METHOD IS REQUIRED
9     List<List<Integer>> ClosestDestinations(List<Integer> numDestinations,
10                                            List<List<Integer>> allocations,
11                                            int numDeliveries)
12    {
13        // WRITE YOUR CODE HERE
14    }
15    // METHOD SIGNATURE ENDS
16 }

```

SUBMIT ANSWER

Amazon Fresh is a grocery delivery service that offers consumers the option of purchasing their groceries online and schedule future deliveries of purchased groceries. Amazon's backend system dynamically tracks each Amazon Fresh delivery truck and automatically assigns the next deliveries in a truck's plan. To accomplish this, the system generates an optimized delivery plan with X destinations. The most optimized plan would deliver to the closest X destinations from the start among all of the possible destinations in the plan.

Given an array of N possible delivery destinations, implement an algorithm to create the delivery plan for the closest X destinations.

Input

The input to the function/method consists of three arguments:

numDestinations, an integer representing the total number of possible delivery destinations for the truck (N);

allLocations, a list where each element consists of a pair of integers representing the x and y coordinates of the delivery locations;

numDeliveries, an integer representing the number of deliveries that will be delivered in the plan (X).

Output

Return a list of elements where each element of the list represents the x and y integer coordinates of the delivery destinations.

Note

The plan starts from the truck's location [0, 0]. The distance of the truck from a delivery destination (x, y) is the square root of $x^2 + y^2$. If there are ties then return any of the locations as long as you satisfy returning X deliveries.

Example

Input:

```
numDestinations = 3  
allLocations = [[1, 2], [3, 4], [1, -1]]  
numDeliveries = 2
```

Output:

```
[[1, -1], [1, 2]]
```

Explanation:

The distance of the truck from location [1, 2] is square root(5) = 2.236

The distance of the truck from location [3, 4] is square root(25) = 5

The distance of the truck from location [1, -1] is square root(2) = 1.414

numDeliveries is 2, hence the output is [1, -1] and [1, 2].

3. Steak House 牛排屋问题

9. Maze 迷宫问题 , removeObstacle

<https://www.1point3acres.com/bbs/thread-469583-1-1.html>

<https://www.1point3acres.com/bbs/thread-457483-1-1.html>

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=468517>

给定一个二维矩阵代表一个有待开发的停车场，1 代表平地，

0 代表沟壑，9 代表障碍物；有个机器人可以在平地上行走，不能进入沟壑，

目的是找到最短距离使得机器人从左上角去到障碍物所在的地方。

我用 BFS，然后记录每个点的最短距离，完成所有搜索后，

再输出障碍物的最短距离。关键是如何 update 最短距离。

做第二题的时候，网络好像不太好，compiler 没有反应，

然后它并不会更新 output 的结果，所以我以为我的更改还是没对，

幸运的是在最后一分钟的时候，再 compile 一遍发现过了。小心出现这种情况。

Automata Amazon

Question 2 out of 2

Problem

Test Cases

Output

You are in charge of preparing a recently purchased lot for one of Amazon's new building. The lot is covered with trenches and has a single obstacle that needs to be taken down before the foundation can be prepared for the building. The demolition robot must remove the obstacle before progress can be made on the building.

Write an algorithm to determine the minimum distance required for the demolition robot to remove the obstacle.

Assumptions:

- The lot is flat, except for trenches, and can be represented as a two-dimensional grid.
- The demolition robot must start from the top-left corner of the lot, which is always flat, and can move one block up, down, left, or right at a time.
- The demolition robot cannot enter trenches and cannot leave the lot.
- The flat areas are represented as 1, areas with trenches are represented by 0 and the obstacle is represented by 9.

Input

The input to the function/method consists of three arguments:

numRows, an integer representing the number of rows;

numColumns, an integer representing the number of columns;

lot, representing the two-dimensional grid of integers.

Output

Return an integer representing the minimum distance traversed to remove the obstacle else return -1.

Constraints

$1 \leq \text{numRows}, \text{numColumns} \leq 1000$

Input

The input to the function/method consists of three arguments:
numRows, an integer representing the number of rows;
numColumns, an integer representing the number of columns;
lot, representing the two-dimensional grid of integers.

Output

Return an integer representing the minimum distance traversed to remove the obstacle else return -1.

Constraints

$1 \leq \text{numRows}, \text{numColumns} \leq 1000$

Example

Input:

numRows = 3

numColumns = 3

lot =

`[[1, 0, 0],`

`[1, 0, 0],`

`[1, 9, 1]]`

Output:

3

Explanation:

Starting from the top-left corner, the demolition robot traversed the cells (0,0) -> (1,0) -> (2,0) -> (2,1).

The robot traversed the total distance 3 to remove the obstacle.

So, the output is 3.

Testcase 1:

Input:

```
3, 3,  
[[1, 0, 0],  
 [1, 0, 0],  
 [1, 9, 1]]
```

Expected Return Value:

```
3
```

Testcase 2:

Input:

```
5, 4,  
[[1, 1, 1, 1],  
 [0, 1, 1, 1],  
 [0, 1, 0, 1],  
 [1, 1, 9, 1],  
 [0, 0, 1, 1]]
```

Expected Return Value:

```
5
```

```
1 // INCLUDE HEADER FILES NEEDED BY YOUR PROGRAM  
2 // SOME LIBRARY FUNCTIONALITY MAY BE RESTRICTED  
3 // DEFINE ANY FUNCTION NEEDED  
4 // FUNCTION SIGNATURE BEGINS, THIS FUNCTION IS REQUIRED  
5 int removeObstacle(int numRows, int numColumns, int **lot)  
6 {  
7     // WRITE YOUR CODE HERE  
8 }  
9 // FUNCTION SIGNATURE ENDS
```

WorkStyle :

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=445121&highlight=amazon%2Boa>

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=474434&highlight=amazon>

比较全的笔记：

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=480213&highlight=oa2>