* 1. **Shortest distance to 9 in matrix**

<https://leetcode.com/problems/the-maze-ii/>

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

o 
o 
no a fully working code over 
unnot review this problem again. You can 
our code. The System.out.print/n() may not work in case of syntaxfruntime 
Il version of JDK being used is 1.8. 
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 flat areas are represented as n, areas with trenches are represented by O and the obstacle is represented by 
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. 
Re'turn an integer repres-enting the minimum distance traversed to remove the obstacle else return -l. 
numRows, numCo/umns 1000 
Example 
Input: 
3 
numRows 
3 
num Columns 
lot 
[1, o, 01, 
.,enut: 
2 'h CLASSES 
3 cass 
inert jav•.util.List•, 
5 v ccass 
6 public class Solutien 
int Ott 
11 
4, 

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

Automata Amazon Question 
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 Hie 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 tie 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 fiat areas are represented as 1, areas with trenches are represented by O and the obstacle 
is represented by g. 
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 tie two-dimensional grid of integers. 
Output 
Return an integer representing the minimum distance traversed to remove the obstacle else retum -1. 
Constraints 
1 numRows, numColumns 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 tie two-dimensional grid of integers. 
Output 
Return an integer representing the minimum distance traversed to remove the obstacle else retum -1. 
Constraints 
1 numRows, numColumns 1000 
Example 
Input: 
numRows = 3 
numColumns = 3 
lot = 
o, 01, 
(1, g, Ill 
Output: 
3 
Explanation: 
Starting from the top-left comer, the demolition robot traversed the cells (0,0) (1 (2,0) -> (2, 1). 
The robot traversed the total distance 3 to remove the obstacle. 
So, the output is 3.

Testcase 1 : 
Input: 
[11, o, 01, 
[1, o, 01, 
[1, g, 111 
Expected Return Value: 
Testcase 2: 
Input: 
[0, 
[0, 
[1, 
[0, 
11, 
11, 
11, 
11, 
Expected Return Value:

* 1. **Longest palindrome substring 看马拉车**

解法： <https://leetcode.com/problems/longest-palindromic-substring/>

<https://www.1point3acres.com/bbs/thread-472789-1-1.html>

# 

* 1. **Most frequent word**

<https://www.1point3acres.com/bbs/thread-472789-1-1.1html>

解法 <https://leetcode.com/problems/most-common-word/>

* 1. **Airplane Optimal Utilization**

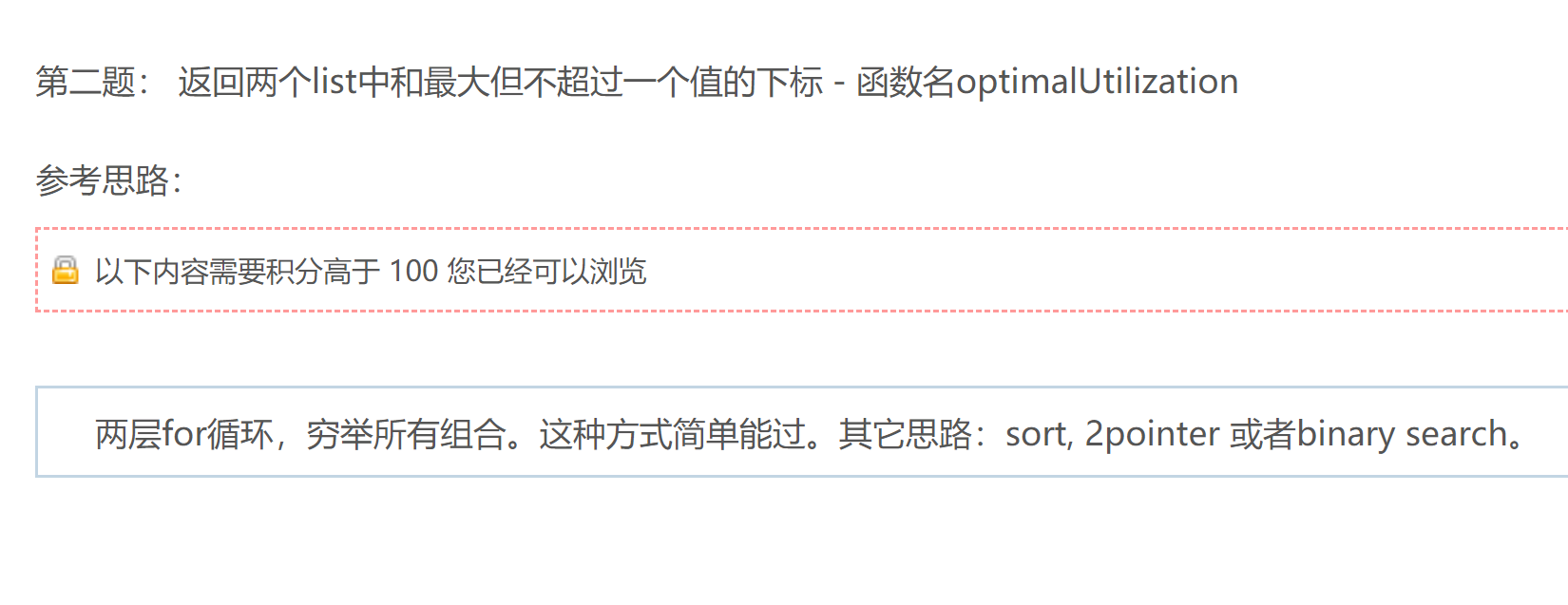
<https://www.1point3acres.com/bbs/thread-459094-1-1.html>

Input 
The input to the function/method consists Of three arguments: 
maxTravelDist, an integer representing the maximum operating travel distance of the given aircraft; 
forwardRouteList, a list of pairs of integers where the first integer represents the unique identifier of a forward 
shipping route and the second integer represents the amount of travel distance required by this shipping route; 
returnRouteList, a list Of pairs Of integers where the first integer represents the unique identifier Of a returned 
shipping route and the second integer represents the amount Of travel distance required by this shipping route; 
Output 
Return a list of pairs of integers representing the pairs of IDs of forward and return shipping routes that optimally 
utilize the given aircraft. If no route is possible, return an empty list 
Example: 
Input: 
max TravelDist = 7000 
forwardRouteList = [(1,2000], [2, 40001, [3, 600011 
returnRouteList 
Output 
pair, 
_üok 
tabl*FFFrEä5distancencoordinate 
mapping, 

<https://www.1point3acres.com/bbs/thread-467307-1-1.html>

Output 
Return a list Of pairs Of integers representing the pairs of IDs of forward and return shipping routes that optimally 
utilize the given aircraft. If no route is possible, return an empty list. 
Examples 
Example 1: 
Input: 
maxTravelDist = 7000 
forwardR0LlteList = [11 
returnRouteust = [[1 
Output: 
111 
Explanation: 
There are only three combinations, [1 ,1], [2, 1], and [3, 1], which have a total of 4000, 6000, and 8000 miles, 
respectively. Since 6000 is the largest use that does not exceed 7000, [2, 1] is the only optimal pair. 
Example 2: 
Input: 
maxTravelDist 10000 
forwardRouteList [11, 3000], [2, 5000], [3, 7000], [4, 10000]] 
returnRouteList [[1 , 2000], [2, 3000], [3, 4000], [4, 5000]] 
Output: 
[P, 41, [3, 21] 
Explanation: 
There are two pairs of forward and return shipping routes possible that optimally utilizes the given aircraft. 
Shipping Route ID#2 from the forwardShippingRouteList requires 5000 miles travelled, and Shipping Route ID#4 
from returnShippingRouteList also requires 5000 miles travelled. Combined, they add up to 10000 miles travelled. 
Similarly, Shipping Route ID#3 from forwardShippingRouteList requires 7000 miles travelled, and Shipping Route 
ID#2 from returnShippingRouteList requires 3000 miles travelled. These also add up to 10000 miles travelled. 
Therefore, the pairs of forward and return shipping routes that optimally utilize the aircraft are [2, 41 and [3, 2].

<https://www.1point3acres.com/bbs/thread-467824-1-1.html>



<https://www.1point3acres.com/bbs/thread-468910-1-1.html>

—EEforward routes, —GEreturn routes, 
search, 
test 
questions , 

<https://www.1point3acres.com/bbs/thread-460321-1-1.html>

背景跟前端個執行一個程式 給定一個記憶體限制 找到可以執行的前後端程式配對最大值    跟飛機送貨是同樣的演算法

3個輸入   integer maxCapacity =>不能超過    兩個vector<pair<int, int>> 一個代表foreground   第一個是id  第二個是capacity of this id   第二個代表background 主要就是把每個element of foreground跟每個element of background一個一個加起來  如果沒超過maxCapacity  就看看是不是超過目前最大值  式的話就記下來  然後return 這個vector

<https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=465736>

 Given two arrays of integers, and a number, find the pair of integers from each array, and the max sum of two integers are equal or less than capacity.

第二题就是给2个sorted array，和一个整数capacity，每个array各找出一个数，组成一个pair。找出pair满足以下条件：

1）sum of pair <= capacity

2) sum is maximum

附加：

Work simulation:

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

<https://www.1point3acres.com/bbs/thread-454772-1-1.html>

<https://www.1point3acres.com/bbs/thread-464257-1-1.html>

public class Solution {

private int lo, maxLen;

public String longestPalindrome(String s) {

int len = s.length();

if (len < 2)

return s;

for (int i = 0; i < len-1; i++) {

extendPalindrome(s, i, i); //assume odd length, try to extend Palindrome as possible

extendPalindrome(s, i, i+1); //assume even length.

}

return s.substring(lo, lo + maxLen);

}

private void extendPalindrome(String s, int j, int k) {

while (j >= 0 && k < s.length() && s.charAt(j) == s.charAt(k)) {

j--;

k++;

}

if (maxLen < k - j - 1) {

lo = j + 1;

maxLen = k - j - 1;

}

}}

public String longestPalindrome(String s) {

int n = s.length();

String res = null;

boolean[][] dp = new boolean[n][n];

for (int i = n - 1; i >= 0; i--) {

for (int j = i; j < n; j++) {

dp[i][j] = s.charAt(i) == s.charAt(j) && (j - i < 3 || dp[i + 1][j - 1]);

if (dp[i][j] && (res == null || j - i + 1 > res.length())) {

res = s.substring(i, j + 1);

}

}

}

return res;

}

23280674918378