

# Assignment #C: bfs & dp

Updated 1436 GMT+8 Nov 25, 2025

2025 fall, Complied by 郭旭杰、化学与分子工程学院

账户：OpenJudge: 25n2500011906, 昵称：郭旭杰

LeetCode/CodeForces/Luogu/sunnywhy/OnlineJudge: LittleBeetroot

## 说明：

- 1) 请把每个题目解题思路（可选），源码Python, 或者C++（已经在Codeforces/Openjudge上AC），截图（包含Accepted），填写到下面作业模版中（推荐使用 typora <https://typoraio.cn>，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。
- 2) 提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、“作业评论”区有上传的md或者doc附件。
- 3) 如果不能在截眼前提交作业，请写明原因。

由于不可抗拒因素(OpenJudge平台无法连接)，本人可能无法按时完成OpenJudge上的题目，本次作业多做三道sunnywhy上的题目作为补偿，落下的OpenJudge题目将于下次提交作业时一同补齐，敬请谅解。



## 1. 题目

### sy321迷宫最短路径

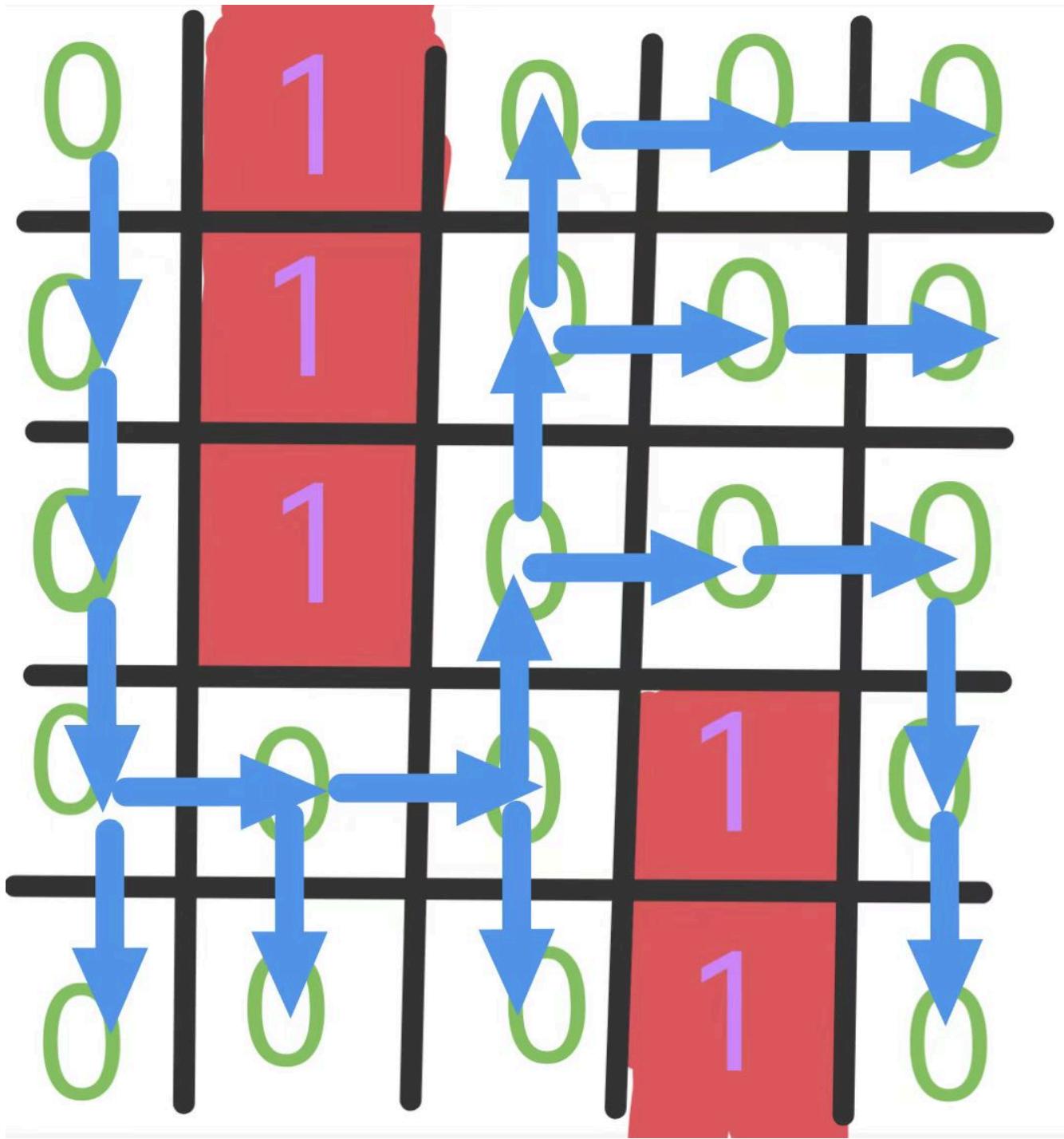
bfs, hashing, <https://sunnywhy.com/sfbj/8/2/321>

耗时：3h

思路：bfs类型的题目对思维能力和抽象能力有较强的要求，以下是我充分思考并查阅有关文献后得到的解题思路：

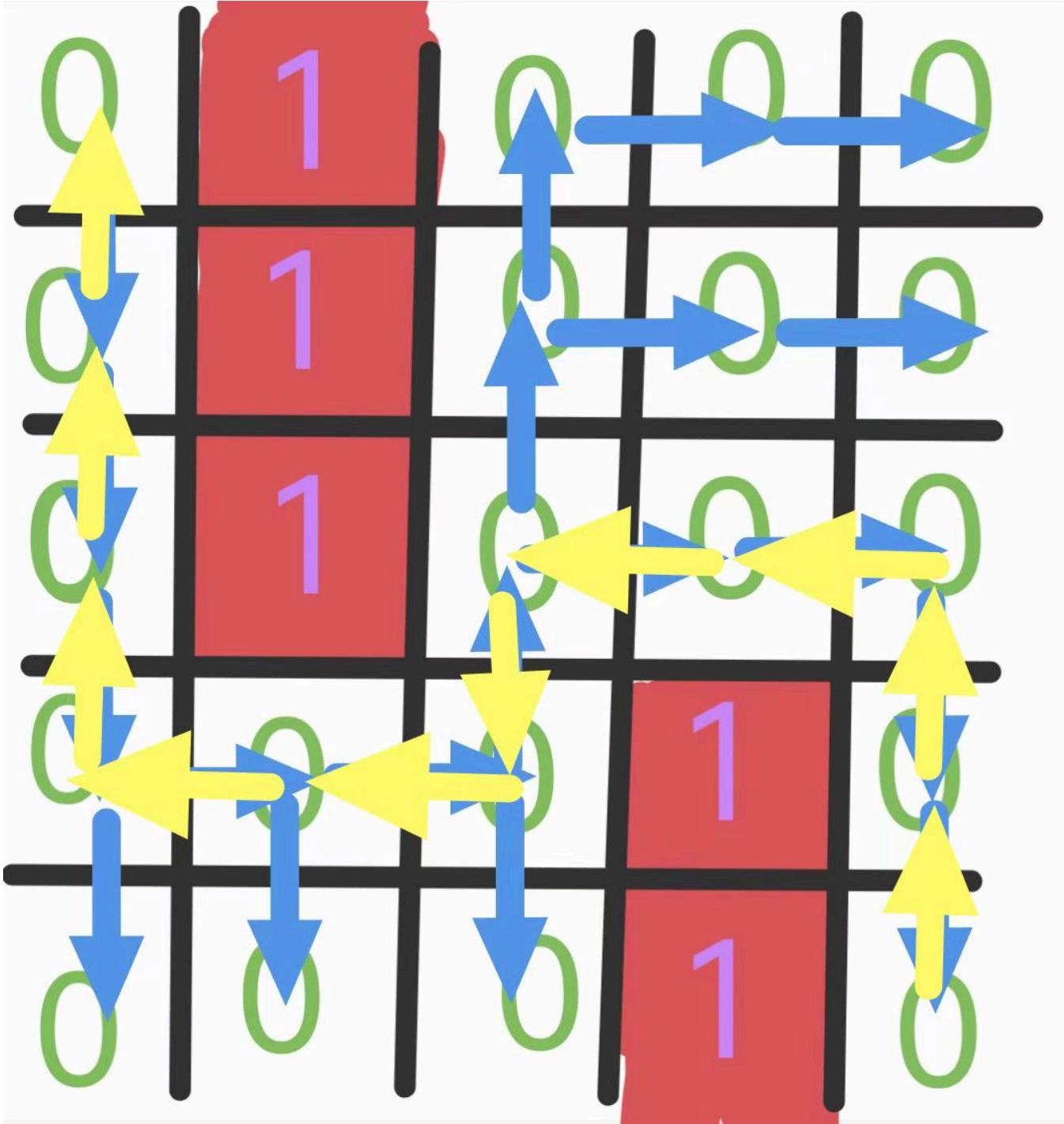
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	0	0	1	0
0	0	0	1	0

如图所示，1为墙壁，0为平地。



我们可以想象有一支由很多人组成的探险队从  $(0,0)$  出发探索未知的迷宫，有些时候有不同的方向可以走就会兵分多路。但是为了避免迷路，他们每次都会用蓝色箭头标记一下自己的行进方向，直到迷宫的每一个可以探索的方格都被探索且仅被探索1次为止。

在我的解法中，我用一个字典(哈希表)来表示所有的蓝色箭头，从而实现回溯过程。



数据确保最终一定会有队员找到位于右下角的迷宫出口，此时他们只要逆着蓝色箭头行进就一定可以回到起点，且由于走最短路线的分队会率先到达并“占领”终点，故其必然走最短路径。

但是从起点顺着蓝色箭头不一定可以到达终点，找到出口的队员为了让位于起点的大部队找到通达终点的最短路径，应该一边往回走一边用如图所示的黄色箭头做标记，确定唯一最短路径(代码中用`ress = res[::-1]`表示)。

最后逆黄色箭头而行即为所求结果，注意行列的序号均要加一。

代码：

```
import sys

dir4 = [(0, 1), (1, 0), (0, -1), (-1, 0)]
```

```
def solve():
    n, m = list(map(int, sys.stdin.readline().split()))
    matrix = [list(map(int, sys.stdin.readline().split())) for _ in range(n)]
    visited = [[False for _ in range(m)] for _ in range(n)]

    fa = {}

    def bfs(i, j):
        if matrix[i][j] == 0 and not visited[i][j]:
            visited[i][j] = True
            for di, dj in dir4:
                if 0 <= i + di <= n - 1 and 0 <= j + dj <= m - 1:
                    if matrix[i + di][j + dj] == 0 and not visited[i + di][j + dj]:
                        fa[(i + di, j + dj)] = (i, j)
                        bfs(i + di, j + dj)

    bfs(0, 0)

    res = [(n - 1, m - 1)]
    while (0, 0) not in res:
        res.append(fa[res[-1]])
    ress = res[::-1]
    for r in ress:
        print(r[0] + 1, r[1] + 1)

if __name__ == '__main__':
    solve()
```

代码运行截图 (至少包含有"Accepted")

《2026考研算法：全程训练营（初试 & 机试）》已经上线：<https://sunnywhy.com/camp/3415>，适合包括『浙大、复旦、上交、华师、中科大计算机&软件』等上机难度院校，也适合『难度友好型』院校。

题目

题解

## 迷宫最短路径

通过数 1629 提交数 3299 难度 中等 显示标签 ☆

## 题目描述

现有一个 $n * m$ 大小的迷宫，其中 1 表示不可通过的墙壁，0 表示平地。每次移动只能向上下左右移动一格，且只能移动到平地上。假设左上角坐标是(1, 1)，行数增加的方向为 $x$ 增长的方向，列数增加的方向为 $y$ 增长的方向，求从迷宫左上角到右下角的最少步数的路径。

## 输入描述

第一行两个整数 $n, m$  ( $2 \leq n \leq 100, 2 \leq m \leq 100$ )，分别表示迷宫的行数和列数；

接下来 $n$ 行，每行 $m$ 个整数（值为 0 或 1），表示迷宫。

## 输出描述

从左上角的坐标开始，输出若干行（每行两个整数，表示一个坐标），直到右下角的坐标。

数据保证最少步数的路径存在且唯一。

## 样例1

输入 复制

3 3  
0 1 0  
n n n

代码编写

```
1 import sys
2
3 dir4 = [(0, 1), (1, 0), (0, -1), (-1, 0)]
4
5
6 def solve():
7     n, m = list(map(int, sys.stdin.readline().split()))
8     matrix = [list(map(int, sys.stdin.readline().split())) for _ in range(n)]
9     visited = [[False for _ in range(m)] for _ in range(n)]
10
11     fa = {}
12
13     def bfs(i, j):
14         if matrix[i][j] == 0 and not visited[i][j]:
15             visited[i][j] = True
16             for di, dj in dir4:
17                 if 0 <= i + di <= n - 1 and 0 <= j + dj <= m - 1:
18                     if matrix[i + di][j + dj] == 0 and not visited[i + di][j + dj]:
19                         fa[(i + di, j + dj)] = (i, j)
20                         bfs(i + di, j + dj)
```

测试输入

运行结果

历史提交

提交时间

结果

时长(ms)

语言

2025-12-02 02:27:38

完美通过

0

Python

查看

2025-12-02 02:26:45

完美通过

0

Python

查看

2025-12-02 02:24:55

完美通过

0

Python

查看

收起面板

运行

提交

## 完美通过

100% 数据通过测试

运行时长: 0 ms

---

语言: Python

```
1 import sys
2
3
4 dir4 = [(0, 1), (1, 0), (0, -1), (-1, 0)]
5
6
7 def solve():
8     n, m = list(map(int, sys.stdin.readline().split()))
9     matrix = [list(map(int, sys.stdin.readline().split())) for _ in range(m)]
10    visited = [[False for _ in range(m)] for _ in range(n)]
11
12    fa = {}
13
14    def bfs(i, j):
15        if matrix[i][j] == 0 and not visited[i][j]:
16            visited[i][j] = True
17            for di, dj in dir4:
18                if 0 <= i + di <= n - 1 and 0 <= j + dj <= m - 1:
19                    if matrix[i + di][j + dj] == 0 and not visited[i + di][j + dj]:
20                        fa[(i + di, j + dj)] = (i, j)
21                        bfs(i + di, j + dj)
22
23    bfs(0, 0)
24
25    res = [(n - 1, m - 1)]
26    while (0, 0) not in res:
27        res.append(fa[res[-1]])
```

# sy324多终点迷宫问题

bfs, <https://sunnywhy.com/sfbj/8/2/324>

耗时: 30min

思路: 在sy320的代码中小修小补就可以了。

代码:

```
import sys
from collections import deque

def kpj(listy):
    return ' '.join(listy)

def matrix_out_print(amatrix):
    for arow in amatrix:
        print(kpj(arow))

dir4 = [(0, 1), (1, 0), (0, -1), (-1, 0)]

def solve():
    def to_get_mig():
        n, m = list(map(int, sys.stdin.readline().split()))
        matrix = [list(map(int, sys.stdin.readline().split())) for _ in range(n)]
        visited = [[False for _ in range(m)] for _ in range(n)]
        mig = [['-1' for _ in range(m)] for _ in range(n)]

        cnt = 0
        queue = deque([(0, 0)])
        visited[0][0] = True

        while queue:
            lq = len(queue)
            for _ in range(lq):
                i, j = queue.popleft()
                mig[i][j] = str(cnt)
                for di, dj in dir4:
                    if 0 <= i + di <= n - 1 and 0 <= j + dj <= m - 1:
                        if matrix[i + di][j + dj] == 0 and not visited[i + di][j + dj]:
                            queue.append((i + di, j + dj))
                            visited[i + di][j + dj] = True
            cnt += 1

        return mig

    matrix_out_print(to_get_mig())
```

```

if __name__ == '__main__':
    solve()

```

## 代码运行截图 (至少包含有"Accepted")

The screenshot shows a programming competition interface. On the left, there's a sidebar with categories like '提高篇 (2)' and '广度优先搜索 (BFS)'. The main area displays a problem titled '多终点迷宫问题' (Multi-End Maze Problem). It includes sections for '题目描述' (Description), '输入描述' (Input Description), '输出描述' (Output Description), and '样例1' (Example 1). The '输出描述' section specifies that the output is a grid of integers representing the minimum steps from the top-left corner to each cell. The '样例1' section shows an input of '3 3' followed by a 3x3 grid of '0's and '1's representing a maze. On the right, a code editor shows the Python solution:

```

1 import sys
2 from collections import deque
3
4
5 def kpj(listy):
6     return ' '.join(listy)
7
8
9 def matrix_out_print(amatrix):
10    for arrow in amatrix:
11        print(kpj(arrow))
12
13
14 dir4 = [(0, 1), (1, 0), (0, -1), (-1, 0)]
15
16
17 def solve():
18     def to_get_mig():
19         n, m = list(map(int, sys.stdin.readline().split()))
20         matrix = [list(map(int, sys.stdin.readline().split())) for _ in range(m)]
21         visited = [[False for _ in range(m)] for _ in range(n)]
22         mig = [[-1 for _ in range(m)] for _ in range(n)]

```

The status bar indicates '完美通过' (Accepted) and '100% 数据通过测试' (All test cases passed).

## 189A. Cut Ribbon

brute force/dp, 1300, <https://codeforces.com/problemset/problem/189/A>

耗时: 30min

思路: 整体上用暴力求解, 算法上属于dp中的简单类型, 但是要进行分类讨论否则当a, b, c中有0的时候会爆栈。

负数对应的base值要足够小(例如:-10000), 否则会WA。

代码:

```

from functools import lru_cache
import sys
sys.setrecursionlimit(180000)

n0, a0, b0, c0 = list(map(int, input().split()))

```

```

@lru_cache(maxsize=None)
def dp(n, a, b, c):
    if n < 0:
        return -10000
    elif n == 0:
        return 0
    else:
        return max(dp(n - a, a, b, c) + 1, dp(n - b, a, b, c) + 1, dp(n - c, a, b, c) + 1)

if n0 % min(a0, b0, c0) == 0:
    print(n0 // min(a0, b0, c0))
else:
    print(dp(n0, a0, b0, c0))

```

代码运行截图 (至少包含有"Accepted")

The screenshot shows a Codeforces contest interface. At the top, there's a navigation bar with links like HOME, TOP, CATALOG, CONTESTS, GYM, PROBLEMSET, GROUPS, RATING, EDU, API, CALENDAR, and HELP. On the right, there are user profile links for "LittleBeetroot" and "Logout". Below the navigation bar, there's a search bar and a "PROBLEMS" dropdown menu.

The main area displays a table of results for a specific problem. The table has columns for # (problem ID), Author (Practice: LittleBeetroot), Problem (189A - 39), Lang (PyPy 3-64), Verdict (Accepted), Time (125 ms), Memory (241560 KB), Sent (2025-12-01 23:42:48), Judged (2025-12-01 23:42:48), and two additional columns for marking and comparing submissions.

Below the table, there's a "Source" tab where the submitted code is displayed:

```

from functools import lru_cache
import sys
sys.setrecursionlimit(180000)

n0, a0, b0, c0 = list(map(int, input().split()))

@lru_cache(maxsize=None)
def dp(n, a, b, c):
    if n < 0:
        return -10000
    elif n == 0:
        return 0
    else:
        return max(dp(n - a, a, b, c) + 1, dp(n - b, a, b, c) + 1, dp(n - c, a, b, c) + 1)

if n0 % min(a0, b0, c0) == 0:
    print(n0 // min(a0, b0, c0))
else:
    print(dp(n0, a0, b0, c0))

```

At the bottom left, there's a link "Click to see test details".

#	When	Who	Problem	Lang	Verdict	Time	Memory
<a href="#">351524019</a>	Dec/02/2025 04:42 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Accepted	125 ms	241600 KB
<a href="#">351523896</a>	Dec/02/2025 04:41 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Wrong answer on test 9	78 ms	236000 KB
<a href="#">351523772</a>	Dec/02/2025 04:39 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Runtime error on test 5	78 ms	237300 KB
<a href="#">351523727</a>	Dec/02/2025 04:38 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Memory limit exceeded on test 5	78 ms	262100 KB
<a href="#">351523702</a>	Dec/02/2025 04:38 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Runtime error on test 5	78 ms	134700 KB
<a href="#">351523682</a>	Dec/02/2025 04:38 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Runtime error on test 5	62 ms	45000 KB
<a href="#">351523621</a>	Dec/02/2025 04:37 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Runtime error on test 5	109 ms	19300 KB
<a href="#">351523327</a>	Dec/02/2025 04:33 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Runtime error on test 5	109 ms	12300 KB
<a href="#">351523295</a>	Dec/02/2025 04:32 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Memory limit exceeded on test 1	46 ms	262100 KB
<a href="#">351523211</a>	Dec/02/2025 04:31 <sup>UTC+8</sup>	LittleBeetroot	<a href="#">A - Cut Ribbon</a>	PyPy 3-64	Runtime error on test 5	78 ms	5500 KB

## M02945:拦截导弹

dp, greedy <http://cs101.openjudge.cn/pctbook/M02945>

耗时: 1h

思路: 直接brute force查询所有子集会爆内存。

应该使用dp, 用ts列表储存从每一个导弹开始拦截所能够拦截的最大导弹数量, 最后输出总共可以拦截的最大导弹数量。

代码:

```

k = int(input())
js = list(map(int, input().split()))
ts = [[j, 1] for j in js]
for i in range(1, k + 1):
    spl = [0]
    for t in range(k - i + 1, k):
        if ts[t][0] <= ts[k - i][0]:
            spl.append(ts[t][1])
    ts[k - i][1] = max(spl) + 1

res = [t[1] for t in ts]
print(max(res))

```

代码运行截图 (至少包含有"Accepted")

**M02945:拦截导弹**[查看](#)[提交](#)[统计](#)[提问](#)

总时间限制: 1000ms 内存限制: 65536kB

**描述**

某国为了防御敌国的导弹袭击，开发出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭，并观测到导弹依次飞来的高度，请计算这套系统最多能拦截多少导弹。拦截来袭导弹时，必须按来袭导弹袭击的时间顺序，不允许先拦截后面的导弹，再拦截前面的导弹。

**输入**

输入有两行，

第一行，输入雷达捕捉到的敌国导弹的数量k ( $k \leq 25$ )，

第二行，输入k个正整数，表示k枚导弹的高度，按来袭导弹的袭击时间顺序给出，以空格分隔。

**输出**

输出只有一行，包含一个整数，表示最多能拦截多少枚导弹。

**样例输入**

```
8
300 207 155 300 299 170 158 65
```

**样例输出**

```
6
```

**来源**

医学部计算概论2006期末考试题

[查看](#) [提交](#) [统计](#) [提问](#)

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)全局题号 **1947**添加于 **2025-03-13**提交次数 **404**尝试人数 **210**通过人数 **209****你的提交记录**

#	结果	时间
7	Accepted	2025-12-02
6	Accepted	2025-12-02
5	Accepted	2025-11-29
4	Memory Limit Exceeded	2025-11-29
3	Memory Limit Exceeded	2025-11-29
2	Memory Limit Exceeded	2025-11-29
1	Memory Limit Exceeded	2025-11-29



## #51100303提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

k = int(input())
js = list(map(int, input().split()))
ts = [[j, 1] for j in js]
for i in range(1, k + 1):
    spl = [0]
    for t in range(k - i + 1, k):
        if ts[t][0] <= ts[k - i][0]:
            spl.append(ts[t][1])
    ts[k - i][1] = max(spl) + 1

res = [t[1] for t in ts]
print(max(res))

```

## 基本信息

#: 51100303  
 题目: M02945  
 提交人:  
 25n2500011906(Little(25n2500011906))  
 内存: 46860kB  
 时间: 126ms  
 语言: PyPy3  
 提交时间: 2025-12-02 15:58:13

## M01384: Piggy-Bank

dp, <http://cs101.openjudge.cn/practice/01384/>

耗时: 2h

思路: dp思路与189A.Cut Ribbon类似, 很有难度。

我自己照着189A.Cut Ribbon的代码来写, 结果一直TLE。

腾讯元宝提供的思路: 这里使用了不断维护和更新dp数组的思路, 如果可以达到, 则dp[w]的值不是正无穷。这样时间复杂度降低到了O(n)。

这种思路并不是像传统方法一样从p,w开刀, 而是运用逆向思维, 反其道而行之; 看似piget比p和w都大, 实则却化虚为实, 化递归为迭代, 变“小而深”为“大而浅”, 非常巧妙。

代码:

```

for _ in range(int(input())):
    e, f = list(map(int, input().split()))
    piget = f - e
    mat = []
    for _ in range(int(input())):
        p, w = list(map(int, input().split()))
        mat.append((p, w))

    dp = [float('inf')] for _ in range(piget + 1)]
    dp[0] = 0

```

```

for p, w in mat:
    for i in range(w, piget + 1):
        dp[i] = min(dp[i], dp[i - w] + p)

res = dp[piget]
if res == float('inf'):
    print("This is impossible.")
else:
    print(f'The minimum amount of money in the piggy-bank is {res}.')

```

代码运行截图 (至少包含有"Accepted")

OpenJudge
题目ID, 标题, 描述
25n2500011906 信箱 账号

CS101 / 题库 (包括计概、数算题目)

题目
排名
状态
提问

01384:Piggy-Bank
查看 提交 统计 提问

总时间限制: 1000ms 内存限制: 65536kB
全局题号 386  
添加于 2021-12-16  
提交次数 687  
尝试人数 187  
通过人数 184

**描述**

Before ACM can do anything, a budget must be prepared and the necessary financial support obtained. The main income for this action comes from Irreversibly Bound Money (IBM). The idea behind is simple. Whenever some ACM member has any small money, he takes all the coins and throws them into a piggy-bank. You know that this process is irreversible, the coins cannot be removed without breaking the pig. After a sufficiently long time, there should be enough cash in the piggy-bank to pay everything that needs to be paid.

But there is a big problem with piggy-banks. It is not possible to determine how much money is inside. So we might break the pig into pieces only to find out that there is not enough money. Clearly, we want to avoid this unpleasant situation. The only possibility is to weigh the piggy-bank and try to guess how many coins are inside. Assume that we are able to determine the weight of the pig exactly and that we know the weights of all coins of a given currency. Then there is some minimum amount of money in the piggy-bank that we can guarantee. Your task is to find out this worst case and determine the minimum amount of cash inside the piggy-bank. We need your help. No more prematurely broken pigs!

**输入**

The input consists of T test cases. The number of them (T) is given on the first line of the input file. Each test case begins with a line containing two integers E and F. They indicate the weight of an empty pig and of the pig filled with coins. Both weights are given in grams. No pig will weigh more than 10 kg, that means  $1 \leq E \leq F \leq 10000$ . On the second line of each test case, there is an integer number N ( $1 \leq N \leq 500$ ) that gives the number of various coins used in the given currency. Following this are exactly N lines, each specifying one coin type. These lines contain two integers each, P and W ( $1 \leq P \leq 50000$ ,  $1 \leq W \leq 10000$ ). P is the value of the coin in monetary units, W is its weight in grams.

#	结果	时间
13	Accepted	2025-12-02
12	Time Limit Exceeded	2025-12-02
11	Time Limit Exceeded	2025-12-02
10	Time Limit Exceeded	2025-12-02
9	Time Limit Exceeded	2025-12-02
8	Wrong Answer	2025-12-02
7	Wrong Answer	2025-12-02
6	Runtime Error	2025-12-02
5	Runtime Error	2025-12-02
4	Accepted	2025-12-02
3	Accepted	2025-12-02
2	Runtime Error	2025-12-02
1	Runtime Error	2025-12-02



## #51102883提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

for _ in range(int(input())):
    e, f = list(map(int, input().split()))
    piget = f - e
    mat = []
    for _ in range(int(input())):
        p, w = list(map(int, input().split()))
        mat.append((p, w))

    dp = [float('inf')] * (piget + 1)
    dp[0] = 0

    for p, w in mat:
        for i in range(w, piget + 1):
            dp[i] = min(dp[i], dp[i - w] + p)

    res = dp[piget]
    if res == float('inf'):
        print("This is impossible.")
    else:
        print(f'The minimum amount of money in the piggy-bank is {res}.')

```

## 基本信息

#: 51102883  
 题目: 01384  
 提交人: 25n2500011906(Little(25n2500011906))  
 内存: 28684kB  
 时间: 398ms  
 语言: PyPy3  
 提交时间: 2025-12-02 17:39:34

## LC136:只出现一次的数字

math, <https://leetcode.cn/problems/single-number/submissions/682165014/>

耗时: 5min

思路: 使用异或^操作, 极大地简化了代码, 节省了运行时间。

代码:

```

class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        n = 0
        for i in nums:
            n = n ^ i
        return n

```

代码运行截图 (至少包含有"Accepted")

**136. 只出现一次的数字**

已解答

给你一个 非空 整数数组 `nums`，除了某个元素只出现一次以外，其余每个元素均出现两次。找出那个只出现了一次的元素。

你必须设计并实现线性时间复杂度的算法来解决此问题，且该算法只使用常量额外空间。

**示例 1：**

输入: `nums = [2,2,1]`  
输出: 1

**示例 2：**

输入: `nums = [4,1,2,1,2]`  
输出: 4

**示例 3：**

输入: `nums = [1]`  
输出: 1

**提示：**

- `1 <= nums.length <= 3 * 104`
- `-3 * 104 <= nums[i] <= 3 * 104`
- 除了某个元素只出现一次以外，其余每个元素均出现两次。

通过 61 / 61 个通过的测试用例 用时: 2 d 5 hrs 35 m 59 s  
LittleBeetroot 提交于 2025.12.02 16:25

面向在校学生的专享特惠  
完成认证享 7 折 Plus 会员，享受更多学业及职业成长帮助

① 执行用时分布  
0 ms | 击败 100.00%  
② 消耗内存分布  
19.14 MB | 击败 93.54%

复杂度分析  
40%

测试用例 > 测试结果  
通过 执行用时: 0 ms  
Case 1 Case 2 Case 3  
输入  
nums = [2,2,1]

## sy318数字操作

math, brute force, dfs <https://sunnywhy.com/sfbj/8/2/318>

耗时: 5min

思路: 逆向思维, 从大数出发, 能除以二就除以二, 除不尽就减一, 直到得到1为止。

代码:

```
n = int(input())
cnt = 0
while n != 1:
    if n % 2 == 0:
        n //= 2
        cnt += 1
    else:
        n -= 1
        cnt += 1

print(cnt)
```

## 代码运行截图 (至少包含有"Accepted")

题目 题解

数字操作

通过数 2431 提交数 5817 难度 简单 显示标签 ☆

题目描述

从整数 1 开始，每轮操作可以选择将上轮结果加 1 或乘 2。问至少需要多少轮操作才能达到指定整数  $n$ 。

输入描述

一个整数  $n$  ( $2 \leq n \leq 10^5$ )，表示需要达到的整数。

输出描述

输出一个整数，表示至少需要的操作轮数。

代码书写

```
1 n = int(input())
2 cnt = 0
3 while n != 1:
4     if n % 2 == 0:
5         n /= 2
6         cnt += 1
7     else:
8         n -= 1
9         cnt += 1
10
11 print(cnt)
12
```

测试输入 提交结果 历史提交

完美通过

100% 数据通过测试 [详情](#)  
运行时长: 0 ms

查看题解

## sy319矩阵中的块

dfs <https://sunnywhy.com/sfbj/8/2/319>

耗时: 30min

思路: 类似OpenJudge上的鱼塘个数问题，都是通过dfs找连通块的个数。

代码:

```
dir4 = [(0, 1), (0, -1), (1, 0), (-1, 0)]
cnt = 0

def visitable(i, j):
    if 0 <= i <= n - 1 and 0 <= j <= m - 1 and matrix[i][j] == 1:
        return True
    else:
        return False

n, m = list(map(int, input().split()))
matrix = [list(map(int, input().split())) for _ in range(n)]
visited = [[False for _ in range(m)] for _ in range(n)]

def dfs(i, j):
    if visitable(i, j) and not visited[i][j]:
        visited[i][j] = True
        for di, dj in dir4:
            dfs(i + di, j + dj)
```

```

for i0 in range(n):
    for j0 in range(m):
        if matrix[i0][j0] == 1 and not visited[i0][j0]:
            cnt += 1
            dfs(i0, j0)

print(cnt)

```

### 代码运行截图 (至少包含有"Accepted")

题目
题解

矩阵中的块
通过数 2005 提交数 3837 难度 简单 显示标签 ★

**题目描述**

给定一个由0和1组成的二维矩阵，矩阵的大小为 $n$ 行 $m$ 列。若两个位置的元素都为1，并且这两个位置在水平方向或垂直方向上相邻（即上下左右相邻），则认为这两个1“连通”。连通关系具有传递性，也就是说，如果位置A与位置B相邻，位置B又与位置C相邻，则位置A与位置C也视为连通。所有连通的1构成一个“块”。

请编写程序计算矩阵中一共有多少个“块”。

**输入描述**

第一行两个整数 $n$ 、 $m$  ( $2 \leq n \leq 100, 2 \leq m \leq 100$ )，分别表示矩阵的行数和列数；接下来 $n$ 行，每行 $m$ 个 0 或 1（用空格隔开），表示矩阵中的所有元素。

**输出描述**

输出一个整数，表示矩阵中“块”的个数。

**样例1**

输入	复制
6 7 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1	

代码书写
Python

```

1 dir4 = [(0, 1), (0, -1), (1, 0), (-1, 0)]
2 cnt = 0
3
4 def visitable(i, j):
5     if 0 <= i <= n - 1 and 0 <= j <= m - 1 and matrix[i][j] == 1:
6         return True
7     else:
8         return False
9
10 n, m = list(map(int, input().split()))
11 matrix = [list(map(int, input().split())) for _ in range(n)]
12 visited = [[False for _ in range(m)] for _ in range(n)]
13
14 def dfs(i, j):
15     if visitable(i, j) and not visited[i][j]:
16         visited[i][j] = True
17         for di, dj in dir4:
18             dfs(i + di, j + dj)
19
20 for i0 in range(n):
21     for j0 in range(m):
22         if matrix[i0][j0] == 1 and not visited[i0][j0]:
23             cnt += 1
24             dfs(i0, j0)
25
26 print(cnt)
27

```

测试输入
提交结果
历史提交

**完美通过** 100% 数据通过测试 [详情](#) 运行时长: 0 ms

收起面板
运行
提交

## sy320迷宫问题

bfs, deque <https://sunnywhy.com/sfbj/8/2/320>

耗时: 2h30min

思路：从起点出发，每次遍历相邻的格子，记录次数cnt，如果到达终点，立即停止遍历，输出cnt；如果队列已经被清空仍无法到达终点，输出-1即可。

第一次使用queue，十分生疏~

代码：

```

import sys
from collections import deque

```

```
dir4 = [(0, 1), (1, 0), (0, -1), (-1, 0)]\n\n\ndef bfs():\n    n, m = list(map(int, sys.stdin.readline().split()))\n    matrix = [list(map(int, sys.stdin.readline().split())) for _ in range(n)]\n    visited = [[False for _ in range(m)] for _ in range(n)]\n\n    cnt = 0\n    queue = deque([(0, 0)])\n    visited[0][0] = True\n\n    while queue:\n        lq = len(queue)\n        for _ in range(lq):\n            i, j = queue.popleft()\n            if i == n - 1 and j == m - 1:\n                return cnt\n            for di, dj in dir4:\n                if 0 <= i + di <= n - 1 and 0 <= j + dj <= m - 1:\n                    if matrix[i + di][j + dj] == 0 and not visited[i + di][j + dj]:\n                        visited[i + di][j + dj] = True\n                        queue.append((i + di, j + dj))\n        cnt += 1\n\n    return -1\n\nprint(bfs())
```

代码运行截图 (至少包含有"Accepted")

The screenshot shows a programming competition interface. On the left, there's a sidebar with a tree view of topics like '提高篇 (2) —— 搜索专题' and various search filters. The main area has tabs for '题目' (Problem) and '题解' (Solution). The problem description for '迷宫问题' (Maze Problem) states: '现有一个n \* m大小的迷宫，其中1表示不可通过的墙壁，0表示平地。每次移动只能向上下左右移动一格，且只能移动到平地上。求从迷宫左上角到右下角的最小步数。' The input description says: '第一行两个整数n、m (2≤n≤100, 2≤m≤100)，分别表示迷宫的行数和列数；' and '接下来n行，每行m个整数 (值为0或1)，表示迷宫。' The output description says: '输出一个整数，表示最小步数。如果无法到达，那么输出-1。' Below the problem statement, there's an example input '3 3' and an output '0 1 0'. On the right, the code editor contains the following Python code:

```
1 import sys
2 from collections import deque
3
4 dir4 = [(0, 1), (1, 0), (0, -1), (-1, 0)]
5
6
7 def bfs():
8     n, m = list(map(int, sys.stdin.readline().split()))
9     matrix = [list(map(int, sys.stdin.readline().split())) for _ in range(n)]
10    visited = [[False for _ in range(m)] for _ in range(n)]
11
12    cnt = 0
13    queue = deque([(0, 0)])
14    visited[0][0] = True
15
16    while queue:
17        lq = len(queue)
18        for _ in range(lq):
19            i, j = queue.popleft()
20            if i == n - 1 and j == m - 1:
21                return cnt
22            for di, dj in dir4:
23                if 0 <= i + di <= n - 1 and 0 <= j + dj <= m - 1:
24                    if matrix[i + di][j + dj] == 0 and not visited[i + di][j + dj]:
25                        visited[i + di][j + dj] = True
26                        queue.append((i + di, j + dj))
27
28    cnt += 1
```

## 2. 学习总结和收获

前几天我想尝试自己把题目做出来，结果被题目吓到了，怎么都没有思路；

最后又是查看题解，又是求助AI，又是运行PythonTutor找代码错误，这几道题做了我一宿。

bfs对我来说还是有很大的难度的，而且很多时候bfs与dfs容易混为一谈。

dp继续对数学高要求，难度总体较高。

但是在做dp题目的时候，最好只要有了一个思路，哪怕是再笨的方法，都要敢于尝试，没准能AC。

做题看看题目标签还是很有用的，但是针对实战情况，灵活地选择策略(greedy, dp, dfs, bfs)需要经验的积累。

如果作业题目简单，有否额外练习题目，比如：OJ“计概2024fall每日选做”、CF、LeetCode、洛谷等网站题目。

我参加了周四的学校周测，最终AC3，OpenJudge上最好Python和Pypy都提交一遍，否则有些题目会出现只有Python能通过或只有Pypy能通过的Bug~。

让我非常有成就感的是我独立完成了第一题dfs和第三题dp的程序编写，以前编写dfs和dp或多或少都要看题解。



## CS101 / 20251127 cs101 Mock Exam (hosted by TA) 已经结束

[题目](#) [排名](#) [状态](#) [统计](#) [提问](#)

比赛已经结束

2025-11-27 15:08:00

开始时间

2025-11-27 17:00:00

结束时间

2025年11月TA出的计概（2025fall-cs101: DS Algo DS-11班）课程模拟考试。

请独立完成，不能通讯，如：不能使用微信、邮件、QQ等工具。

考试期间，请同学只访问OJ，不能访问其他网站，不要查看OJ考试之前自己提交的代码。

考试过程中允许可以带10张A4纸大小的cheat sheet，以及空白草稿纸。

题目编号前面的大写字母，相应表明是 Easy/Medium/Tough 级别。

-----

登录别人的账号即视为违纪甚至作弊。把自己的账号密码告诉别人，被别人登录，也视为违纪甚至作弊。如果考前别人用过你的账号，请立即修改密码。

请把你的昵称改为 25nxxxxx，后面部分是学号。<http://cs101.openjudge.cn/mine>  
有同学昵称24n, 23n, ..., 19n开始也是可以的，学号别错，才能找到你的成绩。

题目ID	标题	通过率	通过人数	尝试人数
✓ E02790	迷宫	89%	48	54
✓ E27301	给植物浇水	93%	53	57
✓ M28969	解密	98%	52	53
— M28970	预测赢家	74%	28	38
M29741	神经网络之国	26%	5	19
T28750	倾斜方块	0%	0	1

37

25n2500011906(Little(25n2500011906)

3 04:32:32 01:24:36 00:10:56 01:37:00  
(-3) (-1) (-2)



题目 排名 状态 统计 提问

## E02790:迷宫

总时间限制: 3000ms 内存限制: 65536kB

### 描述

一天Extense在森林里探险的时候不小心走入了一个迷宫，迷宫可以看成是由 $n * n$ 的格点组成，每个格点只有2种状态，.和#，前者表示可以通行后者表示不能通行。同时当Extense处在某个格点时，他只能移动到东南西北(或者说上下左右)四个方向之一的相邻格点上，Extense想要从点A走到点B，问在不走出迷宫的情况下能不能办到。如果起点或者终点有一个不能通行(为#)，则看成无法办到。

### 输入

第1行是测试数据的组数k，后面跟着k组输入。每组测试数据的第1行是一个正整数n ( $1 \leq n \leq 100$ )，表示迷宫的规模是 $n * n$ 的。接下来是一个 $n * n$ 的矩阵，矩阵中的元素为.或者#。再接下来一行是4个整数ha, la, hb, lb，描述A处在第ha行，第la列，B处在第hb行，第lb列。注意到ha, la, hb, lb全部是从0开始计数的。

### 输出

k行，每行输出对应一个输入。能办到则输出“YES”，否则输出“NO”。

### 样例输入

```
2
3
.##
..#
#..
0 0 2 2
5
.....
###.#
...#..
###..
...#.
0 0 4 0
```

查看

提交

统计

提问

全局题号 1792

提交次数 179

尝试人数 54

通过人数 48

### 你的提交记录

#	结果	时间
4	Accepted	2025-11-27
3	Runtime Error	2025-11-27
2	Runtime Error	2025-11-27
1	Runtime Error	2025-11-27



## CS101 / 20251127 cs101 Mock Exam (hosted by TA) 已经结束

题目 排名 状态 统计 提问

## #51030330提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import sys
dir4 = [(-1, 0), (1, 0), (0, -1), (0, 1)]
res = []

for _ in range(int(sys.stdin.readline())):
    n = int(sys.stdin.readline())
    matrix0 = []
    for _ in range(n):
        row = sys.stdin.readline()
        matrix0.append(row)

    visited = [[False] * n for _ in range(n)]

    ha, la, hb, lb = list(map(int, sys.stdin.readline().split()))
    if matrix0[ha][la] == '#' or matrix0[hb][lb] == '#':
        res.append('NO')
    else:
        def dfs(i, j):
            if (not visited[i][j]) and matrix0[i][j] == '.':
                visited[i][j] = True
                for dx, dy in dir4:
                    if 0 <= i + dx < n and 0 <= j + dy < n:
                        dfs(i + dx, j + dy)

        dfs(ha, la)
        if visited[hb][lb]:
            res.append('YES')
        else:
            res.append('NO')

for r in res:
    print(r)
```

## 基本信息

#: 51030330  
题目: E02790  
提交人:  
25n2500011906(Little(25n2500011906))  
内存: 74112kB  
时间: 265ms  
语言: PyPy3  
提交时间: 2025-11-27 16:32:36



## CS101 / 20251127 cs101 Mock Exam (hosted by TA) 已经结束

[题目](#) [排名](#) [状态](#) [统计](#) [提问](#)

## E27301:给植物浇水

[查看](#)[提交](#)[统计](#)[提问](#)

总时间限制: 1000ms 内存限制: 65536kB

## 描述

Alice 和 Bob 打算给花园里的  $n$  株植物浇水。植物排成一行，从左到右进行标记，编号从 0 到  $n - 1$ ，每一株植物都需要浇特定量的水。Alice 和 Bob 每人有一个水罐，容量分别为  $a$  和  $b$ ，最初是满的。他们按下面描述的方式完成浇水：

每株植物都可以由 Alice 或者 Bob 来浇水。Alice 按从左到右的顺序从植物 0 开始浇水；Bob 按从右到左的顺序从植物  $n - 1$  开始浇水。二人同时开始。

不管植物需要多少水，浇水所耗费的时间都是一样的。

如果没有足够的水完全浇灌下一株植物，他 / 她会立即重新灌满浇水罐，再给下一株植物浇水。不能提前重新灌满水罐。保证  $a$  和  $b$  均大于任意一株植物需要浇水的量。

如果 Alice 和 Bob 到达同一株植物，那么当前水罐中水更多的人会给这株植物浇水。如果他俩水量相同，那么 Alice 会给这株植物浇水。

请你写一个程序，计算两人浇灌所有植物过程中重新灌满水罐的总次数。

## 输入

第一行为三个正整数  $n, a, b$ ，分别表示植物个数，Alice 和 Bob 的水罐容量。 $n \leq 100$

第二行为  $n$  个空格分隔的正整数，表示每株植物需要的浇水量。

## 输出

一个正整数，表示 Alice 和 Bob 完成浇水所需要重新灌满水罐的总次数。

全局题号 **27301**提交次数 **107**尝试人数 **57**通过人数 **53**

Other language verions

English

## 你的提交记录

#	结果	时间
1	Accepted	2025-11-27



## CS101 / 20251127 cs101 Mock Exam (hosted by TA) 已经结束

[题目](#) [排名](#) [状态](#) [统计](#) [提问](#)

## #51028595提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
n, a, b = map(int, input().split())
js = list(map(int, input().split()))
inti = 0
intj = n - 1
cnt = 0
a0 = a
b0 = b
while inti < intj:
    if a0 >= js[inti]:
        a0 -= js[inti]
        inti += 1
    else:
        cnt += 1
        a0 = a
        a0 -= js[inti]
        inti += 1
    if b0 >= js[intj]:
        b0 -= js[intj]
        intj -= 1
    else:
        cnt += 1
        b0 = b
        b0 -= js[intj]
        intj -= 1
if inti == intj:
    c0 = max(a0, b0)
    if c0 >= js[intj]:
        cnt += 0
    else:
        cnt += 1

print(cnt)
```

## 基本信息

#: 51028595  
题目: E27301  
提交人:  
25n2500011906(Little(25n2500011906))  
内存: 3668kB  
时间: 29ms  
语言: Python3  
提交时间: 2025-11-27 15:18:56



## CS101 / 20251127 cs101 Mock Exam (hosted by TA) 已经结束

题目 排名 状态 统计 提问

## M28969:解密

查看

提交

统计

提问

总时间限制: 1000ms 单个测试点时间限制: 100ms 内存限制: 65536kB

## 描述

有一种简单的加密算法，对于一个长度为n的字符串，这个算法将会以第 $(n+1)/2$ （向下取整）个字符为中间轴(最左边的字符算第1个字符)，将该字符写在密文的开头，然后对左半部分按照同样的办法进行加密并写下密文，再对右半部分按照同样的办法进行加密并写下密文。以此类推，直到左右部分为空，即完成加密。

例如，如果要对12345678进行加密，第一步将选择4作为中间轴，将其写在密文开头，然后继续对左右两边（123和5678）分别继续按这个算法处理并写下，我们可以将其记作4[123][5678]（[]代表待加密处理的部分）。

对于左半部分123，中间轴是2，左半部分为1，右半部分为3因此加密结果为213（1的中间轴为1，左右均为空，因此结果为1，而3同理）。

对于右半部分5678，中间轴是6，左半部分为5，右半部分为78，因此加密结果为65[78] → 6578（78的中间轴为7，左半部分为空，右半部分为8，因此得到78）。

简单来说，整个加密过程如下：

12345678 → 4[123][5678] → 42[1][3][5678] → 4213[5678] → 42136[5][78] → 42136578

因此，对12345678的加密结果为42136578。

现在给出一个长度为n( $1 \leq n \leq 50000$ )的由数字构成的字符串，这个字符串是加密后的密文，请你还原出加密前的明文。

全局题号 28969

提交次数 88

尝试人数 53

通过人数 52

Other language verions

English

你的提交记录

#	结果	时间
2	Accepted	2025-11-27
	Time Limit	
1	Exceeded	2025-11-27

## 输入

一行，一个长度为n( $1 \leq n \leq 50000$ )的由数字构成的字符串，代表加密后的密文。

## 输出

一行，一个长度同样为n的字符串，代表解密后的明文。



## CS101 / 20251127 cs101 Mock Exam (hosted by TA) 已经结束

题目 排名 状态 统计 提问

## #51030761提交状态

查看 提交 统计 提问

## 状态: Accepted

## 基本信息

#: 51030761

题目: M28969

提交人: 25n2500011906(Little(25n2500011906)

内存: 3876kB

时间: 33ms

语言: Python3

提交时间: 2025-11-27 16:45:00

## 源代码

```
def solv(strry):
    n0 = len(strry)
    if n0 <= 2:
        return strry
    else:
        new_str1 = strry[1:(n0 + 1) // 2]
        new_str2 = strry[(n0 + 1) // 2:]
        return solv(new_str1) + strry[0] + solv(new_str2)

js = input()
print(solv(js))
```

37	 25n2500011906(Little(25n2500011906)	3	04:32:32	01:24:36 (-3)	00:10:56	01:37:00 (-1)	(-2)
----	---	---	----------	------------------	----------	------------------	------

周五晚上参加CodeForces [Educational Codeforces Round 185 \(Rated for Div. 2\)](#) 最终AC2

7640	Od00daisuki	2	141	+ 00:07	-2	+3 01:44		
7640	yhiwbagah	2	141	+ 00:18	-4	+2 01:43		
7654	 TTO76	2	142	+ 00:15	-15	+5 01:17		
7654	 LittleBeetroot	2	142	+ 00:36	-4	+ 01:46		
7654	stil1	2	142	+ 00:33	+4 01:09			
7654	narrow_wood_bridge	2	142	+ 01:46	-2	+ 00:36		
7654	 xenitus77	2	142	+ 00:14	+4 01:28			

周六晚上参加CodeForces [Codeforces Round 1067 \(Div. 2\)](#) 最终AC1

9362	solidmetal	490		490 00:05				
9362	 Aniksamiul	490		490 00:05		-4		
9362	Kritya11	490		490 00:05	-1			
9362	 LittleBeetroot	490		490 00:05				
9362	 Sayan_Kabery	490		490 00:05				
9362	 ammar007	490		490 00:05		-3		
9362	 Imran_Raza	490		490 00:05	-6			

周日上午又参加了LeetCode 周赛 [第 478 场周赛](#) AC2 得8分 1112名

## 第 478 场周赛 排名

已结束

全国 全球 大模型 1796 | 84 人 AK!

排名	用户名	得分	完成时间	题目 1 (4)	题目 2 (4)	题目 3 (5)	题目 4 (7)
1073	 我	8	00:37:17	00:22:17 *15min	00:21:18		
1	 细菌小子	20	00:11:44	00:04:59	00:06:35	00:09:13	00:11:44
2	 Cranky 6agarin...	20	00:20:18	00:20:18	00:20:01	00:19:23	00:18:50
3	 _baozii_	20	00:22:05	00:01:33	00:02:31	00:04:15	00:22:05
4	 pku_erutan	20	00:24:41	00:01:54	00:03:22	00:06:58	00:24:41

排名	用户名	得分	完成时间	题目 1 (4)	题目 2 (4)	题目 3 (5)	题目 4 (7)
1112	我	8	00:37:17	0:02:21:17 *15min	0:02:21:18		
1101	Mystifying l3ab...	8	00:31:59	0:01:15:59 *20min	0:00:06:47		
1102	Sad Cartwright...	8	00:32:44	0:01:19:32 *5min	0:00:27:44		
1103	circle	8	00:32:48	0:01:21:58 *5min	0:00:27:48		
1104	MasTeRW 🌟	8	00:33:28	0:01:15:29 *10min	0:00:23:28		
1105	Tustart	8	00:34:10	0:00:29:42	0:00:34:10		
1106	sidneylyc	8	00:34:16	0:00:20:25 *10min	0:00:24:16		
1107	刘从云	8	00:34:39	0:00:19:19	0:00:34:39		
1108	Affectionate G...	8	00:34:49	0:00:29:49 *5min	0:00:12:25		
1109	qby	8	00:35:36	0:00:26:42	0:00:35:36		
1110	spx	8	00:36:09	0:00:24:03 *10min	0:00:26:09		
1111	Aiox	8	00:36:46	0:00:27:14	0:00:36:46		
1112	LittleBeetroot	8	00:37:17	0:00:22:17 *15min	0:00:21:18		
1113	字里行间	8	00:38:36	0:00:16:51 *10min	0:00:28:36		
1114	ForenozI	8	00:38:38	0:00:30:24	0:00:38:38		
1115	AT-Fieldless	8	00:39:18	0:00:20:03 *15min	0:00:24:18		
1116	Crazy l3oydUUU	8	00:39:49	0:00:30:30 *5min	0:00:34:49		

### 周日下午参加计概小班课并做题练习

✓	13	醉酒的狱卒	Low	20	40.00%
✓	14	买水果	Mid	7	85.71%
✓	15	幕次方	Low	5	80.00%
✓	21	宾果游戏	Low	16	37.50%
✓	22	密码学	Low	11	81.82%

The screenshot shows a web-based online judge system. At the top, there are tabs for "2025-11-30课程", "oj | Problem List", and "oj | 买水果". The main navigation bar includes "Home", "Problems", "Contests", "Status", "Rank", "About", and a user account "littlebeetroot". Below the navigation, there are dropdowns for "Language: Python3" and "Theme: Solarized Light". The code editor contains the following Python script:

```
1 n, m = list(map(int, input().split()))
2 js = list(map(int, input().split()))
3 js.sort()
4 f0 = []
5 fs = {}
6 ts = []
7 res = []
8 for _ in range(m):
9     fruit = input()
10    if fruit not in f0:
11        f0.append(fruit)
12        fs[fruit] = 1
13    else:
14        fs[fruit] += 1
15 for f in f0:
16     ts.append(fs[f])
17 ts.sort(reverse=True)
18
19 sgm = 0
20 for i in range(len(ts)):
21     sgm += ts[i] * js[i]
22 res.append(sgm)
23
24 sgm = 0
25 js.sort(reverse=True)
26 for i in range(len(ts)):
27     sgm += ts[i] * js[i]
28 res.append(sgm)
29
30 print(res[0], res[1])
```

Below the code editor, there are "Status" and "Accepted" buttons, and an orange "Submit" button. The footer of the page includes "Online Judge Footer", "Powered by OnlineJudge", and "Version: 20210928-acce7".

Sample Input 1

```
ILOVEJIGAI  
AWSL
```

Sample Output 1

```
IHGGEFARAE
```

### Hint

注意密钥K的长度可以小于明文长度，此时需要对密钥进行循环

The screenshot shows another instance of an online judge interface. The top navigation and theme settings are identical. The code editor contains the following Python script:

```
1 ori = input()
2 key = input()
3 print(''.join([chr((ord(ori[i]) - 65) + (ord(key[i % len(key)]) - 65)) % 26 + 65) for i in range(len(ori))]))
```

Below the code editor, there are "Status" and "Accepted" buttons, and an orange "Submit" button. The footer of the page includes "Online Judge Footer", "Powered by OnlineJudge", and "Version: 20210928-acce7".