

Assignment #D: Mock Exam下元节

Updated 1729 GMT+8 Dec 4, 2025

2025 fall, Compiled by 郭旭杰、化学与分子工程学院

OpenJudge账号: 25n2500011906, 昵称: 郭旭杰

说明:

1. Dec月考: AC2 (请改为同学的通过数)。考试题目都在“题库(包括计概、数算题目)”里面, 按照数字题号能找到, 可以重新提交。作业中提交自己最满意版本的代码和截图。
2. 解题与记录: 对于每一个题目, 请提供其解题思路(可选), 并附上使用Python或C++编写的源代码(确保已在OpenJudge, Codeforces, LeetCode等平台上获得Accepted)。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。(推荐使用Typora <https://typoraio.cn> 进行编辑, 当然你也可以选择Word。)无论题目是否已通过, 请标明每个题目大致花费的时间。
3. 提交安排: 提交时, 请首先上传PDF格式的文件, 并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的本人头像, 提交的文件为PDF格式, 并且“作业评论”区包含上传的.md或.doc附件。
4. 延迟提交: 如果你预计无法在截止日期前提交作业, 请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。

请按照上述指导认真准备和提交作业, 以保证顺利完成课程要求。

比赛已经结束

2025-12-04 15:08:00

开始时间

2025-12-04 17:00:00


结束时间

2025年12月计概（2025fall-cs101: 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
✓	E29945	神秘数字的宇宙旅行	100%	131	131
—	E29946	删数问题	69%	81	118
✓	E30091	缺德的图书馆管理员	89%	102	114
—	M27371	Playfair密码	65%	50	77
—	T30201	旅行售货商问题	25%	7	28
—	T30204	小P的LLM推理加速	34%	10	29

94		25n2500011906(Little(25n2500011906))	2	01:15:50	00:02:03 (-1)	00:33:47 (-10)	(-1)	(-2)	(-6)	(-2)
----	---	--------------------------------------	---	----------	------------------	-------------------	------	------	------	------

1. 题目

E29945:神秘数字的宇宙旅行

implementation, <http://cs101.openjudge.cn/practice/29945>

耗时: 2min

思路：会f' 格式化输出，注意除号的书写、不要忘记最后输出“End”就没有问题。

代码

```
n = int(input())
while n != 1:
    if n % 2 != 0:
        print(f'{n}*3+1={n*3+1}')
        n = n * 3 + 1
    else:
        print(f'{n}/2={n//2}')
        n = n//2
print('End')
```

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 / 20251204 cs101 Mock Exam下元节 已经结束

题目

排名

状态

统计

提问

E29945:神秘数字的宇宙旅行

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

在数字宇宙中, 每个数字都是一个神秘的旅行者。它们遵循着古老的宇宙法则进行空间跳跃:

如果当前位于偶数维度, 就会跳转到当前维度的一半
如果当前位于奇数维度, 就会跳跃到3倍当前维度再加1的新维度

每个数字旅行者最终都会回到宇宙的起源点——维度1。输入一个正整数n ($n \leq 2,000,000$), 表示旅行者的起始维度, 请输出该旅行者的跳跃轨迹。

输入

一个正整数 n。 ($n \leq 2,000,000$)

全局题号 29945

提交次数 155

尝试人数 131

通过人数 131

Other language versions

English

你的提交记录

#	结果	时间
3	Accepted	2025-12-04
2	Accepted	2025-12-04
1	Wrong Answer	2025-12-04

OpenJudge

题目ID, 标题, 描述

25n2500011906 信箱 账号

CS101 / 20251204 cs101 Mock Exam下元节 已经结束

题目

排名

状态

统计

提问

#51131823提交状态

查看

提交

统计

提问

状态: Accepted

源代码

```
n = int(input())
while n != 1:
    if n % 2 != 0:
        print(f'{n}*3+1={n*3+1}')
        n = n * 3 + 1
    else:
        print(f'{n}/2={n//2}')
        n = n//2
print('End')
```

基本信息

#: 51131823

题目: E29945

提交人: 25n2500011906(Little(25n2500011906))

内存: 3632kB

时间: 24ms

语言: Python3

提交时间: 2025-12-04 15:10:08

E29946:删数问题

monotonic stack, greedy, <http://cs101.openjudge.cn/practice/29946>

耗时: 3h

思路: 使用单调栈, 后一个数字比前一个数字大就从栈顶弹出, 直至用完所有次数找到最小的数。

代码

```
n = input()
k = int(input())

stack = []
for i in n:
    while stack and i < stack[-1] and k > 0:
        stack.pop()
        k -= 1
    stack.append(i)

if k > 0:
    print(int(''.join(stack[:-k])))
else:
    print(int(''.join(stack)))
```

代码运行截图 (至少包含有"Accepted")



CS101 / 题库（包括计概、数算题目）

题目 排名 状态 提问

29946:删数问题

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

键盘输入一个高精度的正整数 n（不超过 250 位），去掉其中任意 k 个数字后剩下的数字按原左右次序将组成一个新的非负整数。编程对给定的 n 和 k，寻找一种方案使得剩下的数字组成的新数最小。

输入

输入两行正整数。

第一行输入一个高精度的正整数 n。

第二行输入一个正整数 k，表示需要删除的数字个数。

输出

输出一个整数，最后剩下的最小数。

样例输入

```
175438
4
```

样例输出

```
13
```

提示

monotonic stack, greedy

全局题号 29946

添加于 2025-12-04

提交次数 404

尝试人数 122

通过人数 115

Other language verions

English

你的提交记录

#	结果	时间
7	Accepted	2025-12-09
6	Accepted	2025-12-09
5	Time Limit Exceeded	2025-12-09
4	Time Limit Exceeded	2025-12-09
3	Compile Error	2025-12-09
2	Time Limit Exceeded	2025-12-04
1	Time Limit Exceeded	2025-12-04

#51201248提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```
n = input()
k = int(input())

stack = []
for i in n:
    while stack and i < stack[-1] and k > 0:
        stack.pop()
        k -= 1
    stack.append(i)

if k > 0:
    print(int(''.join(stack[:-k])))
else:
    print(int(''.join(stack)))
```

基本信息

#: 51201248
题目: 29946
提交人: 25n2500011906(Little(25n2500011906))
内存: 3624kB
时间: 27ms
语言: Python3
提交时间: 2025-12-09 04:38:11

E30091:缺德的图书馆管理员

greedy, <http://cs101.openjudge.cn/practice/30091>

耗时: 5min

思路: 题干描述的类似高中时候做过的数学题“士兵过桥”问题, 属于经典数学题的第一问难度, 用所有人分别与走廊两端的距离的最大值和最小值取代时间的最大值和最小值即可。

代码

```
l = int(input())
n = int(input())
js = list(map(int, input().split()))
ks = []
for j in js:
    if j <= l // 2:
        ks.append(j)
    else:
        ks.append(l + 1 - j)
print(max(ks), l + 1 - min(ks))
```

代码运行截图 (至少包含有"Accepted")

E30091:缺德的图书馆管理员

查看

提交

统计

提问

总时间限制: 1000ms 单个测试点时间限制: 10ms 内存限制: 65536kB

描述

期末考试周已经进入到紧要时间。你是图书馆管理员，正在协助同学们复习备考。图书馆的自习室座位像独木桥一样的紧张。你希望找些乐趣，于是让一些同学到图书馆的"独木桥"走廊上背诵课文，而你留在走廊入口观察同学们。同学们十分无奈，因为这条走廊十分狭窄，只能容纳 1 个人通过。假如有 2 个同学相向而行在走廊上相遇，那么他们 2 个人将无法绕过对方，两人都要原地同时转身，继续行走。但是，可以有多个同时呆在同一个位置。

突然，你收到图书馆闭馆的通知，闭馆铃声即将响起！为了按时离馆，走廊上的同学必须全部撤离。走廊的长度为 L ，同学们只能站在坐标为整数的地方。所有同学的速度都为 1，但一个同学某一时刻来到了坐标为 0 或 $L+1$ 的位置，他就离开了走廊。

每个同学都有一个初始面对的方向，他们会以匀速朝着这个方向行走，中途不会自己改变方向。但是，如果两个同学面对面相遇，他们无法彼此通过对方，于是就分别转身，继续行走。转身不需要任何的时间。

全局题号 **30091**

提交次数 **201**

尝试人数 **114**

通过人数 **102**

Other language versions

[English](#)

你的提交记录

#	结果	时间
2	Accepted	2025-12-04
1	Time Limit Exceeded	2025-12-04

#51132482提交状态

查看

提交

统计

提问

状态: **Accepted**

源代码

```
l = int(input())
n = int(input())
js = list(map(int, input().split()))
ks = []
for j in js:
    if j <= l // 2:
        ks.append(j)
    else:
        ks.append(l + 1 - j)
print(max(ks), l + 1 - min(ks))
```

基本信息

#: 51132482
题目: E30091
提交人: 25n2500011906(Little(25n2500011906))
内存: 4080kB
时间: 36ms
语言: Python3
提交时间: 2025-12-04 15:41:47

M27371:Playfair密码

simulation, string, matrix, implementon <http://cs101.openjudge.cn/practice/27371>

耗时: 4h

思路: 自己用了矩阵和哈希表进行处理，结果样例都能过，却一直WA。

后来就自己打了一遍上课题解的代码，学会了诸如replace()函数和pos_map()函数的用法。

代码

能AC的代码:

```
import sys

def solve():
    input_data = sys.stdin.read().split()
    if not input_data:
        return

    key_input = input_data[0]
    n = int(input_data[1])
    plaintexts = input_data[2:]

    # 使用replace函数把'j'全部转换为'i'
    key = key_input.lower().replace('j', 'i')
    alpha = 'abcdefghijklmnopqrstuvwxyz'
    matrix_chars = []
    seen = set()

    for char in key:
        if char not in seen:
            seen.add(char)
            matrix_chars.append(char)

    for char in alpha:
        if char not in seen:
            seen.add(char)
            matrix_chars.append(char)

    matrix = [matrix_chars[i: i + 5] for i in range(0, 25, 5)]
    pos_map = {char: (i // 5, i % 5) for i, char in enumerate(matrix_chars)}

    def encrypt_text(text):
        # 使用replace函数把'j'全部转换为'i'
        text = text.lower().replace('j', 'i')
        res = []
        i = 0
        length = len(text)

        while i < length:
            a = text[i]
            b = ''

            if i + 1 < length:
                b = text[i + 1]

            if a == b:
                b = 'q' if a == 'x' else 'x'
                i += 1
            else:
                i += 2
        else:
            res.append(a)
```



```

        b = 'q' if a == 'x' else 'x'
        i += 1

    # 使用pos_map获取行数和列数
    r1, c1 = pos_map[a]
    r2, c2 = pos_map[b]
    if r1 == r2:
        res.append(matrix[r1][(c1 + 1) % 5])
        res.append(matrix[r2][(c2 + 1) % 5])
    elif c1 == c2:
        res.append(matrix[(r1 + 1) % 5][c1])
        res.append(matrix[(r2 + 1) % 5][c2])
    else:
        res.append(matrix[r1][c2])
        res.append(matrix[r2][c1])

    return ''.join(res)

for text in plaintexts:
    print(encrypt_text(text))

if __name__ == '__main__':
    solve()

```

自己写的代码：

```

def k0j(listy):
    return ''.join(listy)

key = input()

alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
kg = []
s1 = {}
itt = 0
ress = []
for ki in key:
    if ki == 'j' and 'i' not in s1:
        s1['i'] = [itt // 5, itt % 5]
        kg.append('i')
        itt += 1
    elif ki not in s1:
        s1[ki] = [itt // 5, itt % 5]
        kg.append(ki)
        itt += 1
for ai in alpha:
    if ai not in s1:
        s1[ai] = [itt // 5, itt % 5]
        kg.append(ai)

```

```

        itt += 1

for _ in range(int(input())):
    tara = input()
    res = []
    tof = len(tara)
    itig = 0
    while itig < tof - 1:
        if tara[itig] != tara[itig + 1]:
            a = tara[itig]
            b = tara[itig + 1]
            itig += 2
        else:
            if tara[itig] == 'x':
                a = tara[itig]
                b = 'q'
                itig += 1
            else:
                a = tara[itig]
                b = 'x'
                itig += 1

        if a == 'j':
            a = 'i'
        if b == 'j':
            b = 'i'

        if sl[a][0] == sl[b][0]:
            res.append(kg[(sl[a][0] * 5 + (sl[a][1] + 1) % 5) % 25])
            res.append(kg[(sl[b][0] * 5 + (sl[b][1] + 1) % 5) % 25])
        elif sl[a][1] == sl[b][1]:
            res.append(kg[((sl[a][0] + 1) % 5 * 5 + sl[a][1]) % 25])
            res.append(kg[((sl[b][0] + 1) % 5 * 5 + sl[b][1]) % 25])
        else:
            res.append(kg[(sl[a][0] * 5 + sl[b][1]) % 25])
            res.append(kg[(sl[b][0] * 5 + sl[a][1]) % 25])

    if itig == tof - 1:
        a = tara[itig]
        if a == 'x':
            b = 'q'
        else:
            b = 'x'

        if sl[a][0] == sl[b][0]:
            res.append(kg[(sl[a][0] * 5 + (sl[a][1] + 1) % 5) % 25])
            res.append(kg[(sl[b][0] * 5 + (sl[b][1] + 1) % 5) % 25])
        elif sl[a][1] == sl[b][1]:
            res.append(kg[((sl[a][0] + 1) % 5 * 5 + sl[a][1]) % 25])
            res.append(kg[((sl[b][0] + 1) % 5 * 5 + sl[b][1]) % 25])
        else:
            res.append(kg[(sl[a][0] * 5 + sl[b][1]) % 25])
            res.append(kg[(sl[b][0] * 5 + sl[a][1]) % 25])

```

```
ress.append(k0j(res))

for res in ress:
    print(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

27371:Playfair密码

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

Playfair密码是一种古典加密技术，它使用一个矩阵填充密钥和字母表中的其余字母，通常j被忽略，并被视为i。加密过程涉及将明文分成字母对，并对每对字母根据它们在矩阵中的位置进行加密，同样j也被视为i。

1. 密钥矩阵的生成

1.1 选择密钥：首先选择一个密钥，通常是一个单词或短语，如keyword。

1.2 填充矩阵：
去除重复字母：在密钥中去除重复出现的字母，只保留该重复字母出现的第一次进行填入。
填充密钥：将处理过的密钥填入矩阵的前几个格子，从左至右，从上至下填充。
填充剩余字母：在矩阵中填入字母表的其余字母。在传统Playfair密码中，通常将字母"j"视为字母"i"，以适应5x5矩阵的大小。

2. 加密过程

2.1 准备明文：将明文分成字母对。如果相邻字母相同就在该组的第一个字母后加入"x"后重新分组（如果第一个字母为"x"则加"q"）。如果明文处理后为奇数就在最后加入"x"（若最后一个字母为"x"则加"q"）。

2.2 对每个字母对进行加密：
同一行不同列：如果字母对位于矩阵的同一行，则每个字母都被替换为右侧的字母（最右边的字母替换为最左边的）。
同一列不同行：如果字母对位于矩阵的同一列，则每个字母都被替换为下方的字母（最下面的字母替换为最上面的）。
不同行和列：对于不在同一行或同一列的字母对，每个字母被替换为同一行中与另一个字母所在列相对应的字母。

3.示例：

全局题号 27371

添加于 2025-12-04

提交次数 408

尝试人数 114

通过人数 108

Other language versions

English

你的提交记录

#	结果	时间
29	Accepted	2025-12-09
28	Accepted	2025-12-09
27	Runtime Error	2025-12-09
26	Runtime Error	2025-12-09
25	Wrong Answer	2025-12-09
24	Wrong Answer	2025-12-09
23	Wrong Answer	2025-12-09
22	Wrong Answer	2025-12-09
21	Wrong Answer	2025-12-09
20	Wrong Answer	2025-12-09
19	Time Limit Exceeded	2025-12-09
18	Time Limit Exceeded	2025-12-09
17	Compile Error	2025-12-09
16	Wrong Answer	2025-12-09
15	Wrong Answer	2025-12-09
14	Wrong Answer	2025-12-09
13	Wrong Answer	2025-12-09



CS101 / 题库（包括计概、数算题目）

题目

排名

状态

提问

#51212733提交状态

查看

提交

统计

提问

状态: Accepted

源代码

```
import sys

def solve():
    input_data = sys.stdin.read().split()
    if not input_data:
        return

    key_input = input_data[0]
    n = int(input_data[1])
    plaintexts = input_data[2:]

    # 使用replace函数把'j'全部转换为'i'
    key = key_input.lower().replace('j', 'i')
    alpha = 'abcdefghijklmnopqrstuvwxyz'
    matrix_chars = []
    seen = set()

    for char in key:
        if char not in seen:
            seen.add(char)
            matrix_chars.append(char)

    for char in alpha:
        if char not in seen:
            seen.add(char)
            matrix_chars.append(char)

    matrix = [matrix_chars[i: i + 5] for i in range(0, 25, 5)]
    pos_map = {char: (i // 5, i % 5) for i, char in enumerate(matrix_chars)}

    def encrypt_text(text):
        # 使用replace函数把'j'全部转换为'i'
        text = text.lower().replace('j', 'i')
        res = []
        i = 0
```

基本信息

#: 51212733

题目: 27371

提交人:

25n2500011906(Little(25n2500011906))

内存: 3756kB

时间: 27ms

语言: Python3

提交时间: 2025-12-09 20:17:07

```
T30204.py M27371.py M27371_2.py x M27371_1.py T30201.py T30201_1.py E299.py

1 def k0j(listy):
2     return ''.join(listy)
3
4
5 key = input()
6
7 alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
8         'x', 'y', 'z']
9 kg = []
10 sl = {}
11 itt = 0
12 res = []
13 for ki in key:
14     if ki == 'j' and 'i' not in sl:
15         sl['i'] = [itt // 5, itt % 5]
16         kg.append('i')
17         itt += 1

M27371_2 x

D:\Python311\python.exe "C:\Users\Lenovo\Desktop\code (1)\codes (1)\assignments\assignment_D\M27371_2.py"
keyword
1
balloon
cbizscs

进程已结束，退出代码为 0
```

```
1 def k0j(listy):
2     return ''.join(listy)
3
4
5 key = input()
6
7 alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
8         'x', 'y', 'z']
9 kg = []
10 sl = {}
11 itt = 0
12 res = []
13 for ki in key:
14     if ki == 'j' and 'i' not in sl:
15         sl['i'] = [itt // 5, itt % 5]
16         kg.append('i')
17         itt += 1

M27371_2 x

D:\Python311\python.exe "C:\Users\Lenovo\Desktop\code (1)\codes (1)\assignments\assignment_D\M27371_2.py"
java
3
likejava
programming
aaxx
kaldavbv
qsnhqvrkvsn
vwvwr

进程已结束，退出代码为 0
```

T30201:旅行售货商问题

dp,dfs, <http://cs101.openjudge.cn/practice/30201>

耗时: 2h

思路: 暴力枚举不是TLE就是MLE

AI提供的思路, 使用了状态压缩dp, 要求我们对二进制和位运算能够灵活掌握, 有难度, 但是代码将时间复杂度和空间复杂度由 $O(n!)$ 减小至 $O(n * 2^{**} n)$, 在OpenJudge的Pypy3环境运行能AC。

代码

```
n = int(input())
res = float('inf')
matrix = [list(map(int, input().split())) for _ in range(n)]

dp = [[float('inf')] * n for _ in range(1 << n)]
dp[1 << 0][0] = 0

for mask in range(1 << n):
    for u in range(n):
        if not (mask & (1 << u)):
            continue
        for v in range(n):
            if mask & (1 << v):
                continue
            new_mask = mask | (1 << v)

            if dp[new_mask][v] > dp[mask][u] + matrix[u][v]:
                dp[new_mask][v] = dp[mask][u] + matrix[u][v]

full_mask = (1 << n) - 1
for u in range(n):
    res = min(res, dp[full_mask][u] + matrix[u][0])

print(res)
```

代码运行截图 (至少包含有"Accepted")

30201:旅行售货商问题

查看

提交

统计

提问

总时间限制: 2000ms 内存限制: 262144kB

描述

一个国家有 n 个城市，每两个城市之间都开设有航班，从城市 i 到城市 j 的航班价格为 $\text{cost}[i, j]$ ，而且往、返航班的价格相同。

售货商要从一个城市出发，途径每个城市 1 次（且每个城市只能经过 1 次），最终返回出发地，而且他的交通工具只有航班，请求出他旅行的最小开销。

输入

输入的第 1 行是一个正整数 n ($3 \leq n \leq 18$)
然后有 n 行，每行有 n 个正整数，构成一个 $n * n$ 的矩阵，矩阵的第 i 行第 j 列为城市 i 到城市 j 的航班价格。 $1 \leq \text{cost}[i, j] \leq 10^4$

输出

输出数据为一个正整数 m ，表示旅行售货商的最小开销

样例输入

```
4
0 4 1 3
4 0 2 1
1 2 0 5
3 1 5 0
```

样例输出

```
7
```

全局题号 **30201**
添加于 **2025-12-04**
提交次数 **371**
尝试人数 **97**
通过人数 **80**

你的提交记录

#	结果	时间
6	Accepted	2025-12-09
5	Time Limit Exceeded	2025-12-09
4	Accepted	2025-12-09
3	Memory Limit Exceeded	2025-12-04
2	Memory Limit Exceeded	2025-12-04
1	Memory Limit Exceeded	2025-12-04

#51211023提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```
n = int(input())
res = float('inf')
matrix = [list(map(int, input().split())) for _ in range(n)]

dp = [[float('inf')] * n for _ in range(1 << n)]
dp[1 << 0][0] = 0

for mask in range(1 << n):
    for u in range(n):
        if not (mask & (1 << u)):
            continue
        for v in range(n):
            if mask & (1 << v):
                continue
            new_mask = mask | (1 << v)

            if dp[new_mask][v] > dp[mask][u] + matrix[u][v]:
                dp[new_mask][v] = dp[mask][u] + matrix[u][v]

full_mask = (1 << n) - 1
for u in range(n):
    res = min(res, dp[full_mask][u] + matrix[u][0])

print(res)
```

基本信息

#: 51211023
题目: 30201
提交人: 25n2500011906(Little(25n2500011906))
内存: 104848kB
时间: 1058ms
语言: PyPy3
提交时间: 2025-12-09 18:53:29

T30204:小P的LLM推理加速

greedy, <http://cs101.openjudge.cn/practice/30204>

<https://www.luogu.com.cn/problem/P14635>

耗时: 2h

思路: 常规贪心, 先尽可能购买“便宜”的第一块糖果 (价格不大于最便宜的一组糖果的总价的一半), 再用剩下的钱全部购买最便宜的一组糖果即可。

OpenJudge上AC不成问题, 洛谷上数据只能部分通过, 洛谷官方题解全部是C和C++语言, 我也无法知道洛谷上没有通过的题例是什么样的数据。

代码

```
n, m = list(map(int, input().split()))
min_circ = float('inf')
birds = []
asq = []
for _ in range(n):
    i, j = list(map(int, input().split()))
```



```

birds.append(i)
if i + j < min_circ:
    min_circ = i + j
    asq.append(max(i, j))

early_birds = []
late_birds = [asq[-1]]
for i in birds:
    if i <= min_circ // 2:
        early_birds.append(i)
    elif min_circ // 2 < i <= asq[-1]:
        late_birds.append(i)
early_birds.sort()
late_birds.sort()

cnt = 0
sgm = 0
s0 = sum(early_birds)
l0 = len(early_birds)
if m == s0:
    print(l0)
elif m < s0:
    init_eb = 0
    while sgm <= m - early_birds[init_eb]:
        sgm += early_birds[init_eb]
        init_eb += 1
    print(init_eb)
else:
    m -= s0
    cnt += l0
    cnt += 2 * m // min_circ
    m = m % min_circ
    print(cnt)

```

代码运行截图 (至少包含有"Accepted")

30204:小P的LLM推理加速

查看

提交

统计

提问

总时间限制: 1000ms 内存限制: 262144kB

描述

随着大语言模型参数量的爆炸式增长，边缘设备上的推理效率成为了研究热点。小P的团队正在为一款搭载了新型 NPU（神经网络处理单元）的嵌入式设备开发推理框架。该 NPU 采用了一种特殊的双缓冲机制来管理显存与计算单元的数据交互。

该 NPU 包含 n 个异构的计算核，编号为 1 到 n 。由于硬件架构的特性，每个计算核在执行连续的推理任务时，其 **能耗** 会呈现周期性变化。

具体来说，对于第 i ($1 \leq i \leq n$) 个计算核，其数据加载与计算遵循“冷-热”交替模式：

第 1 个 推理周期（冷启动）：需要从主存加载权重，能耗为 x_i 焦耳。

第 2 个 推理周期（热运行）：权重已在缓存中，直接计算，能耗为 y_i 焦耳。

第 3 个 推理周期（强制刷新）：由于双缓冲队列的限制，缓存被新的数据流覆盖，需要重新加载，能耗变回 x_i 焦耳。

第 4 个 推理周期（热运行）：能耗再次变为 y_i 焦耳。

以此类推

设备的电池总能量预算为 m 焦耳。作为框架的调度算法设计者，你的任务并不关心具体使用了哪些计算核，而是要通过合理分配任务，使得在电量耗尽前，设备总共能完成的 **推理周期数** 最大化。

输入

输入的第一行包含两个正整数 n 和 m ，分别代表计算核的数量和电池的总能量预算。

输入的第 $i+1$ 行 ($1 \leq i \leq n$) 包含两个正整数 x_i 和 y_i ，分别表示第 i 个计算核在“冷启动”“热运行”状态下的能耗。

全局题号 30204

添加于 2025-12-04

提交次数 357

尝试人数 99

通过人数 77

你的提交记录

#	结果	时间
7	Accepted	2025-12-09
6	Accepted	2025-12-09
5	Wrong Answer	2025-12-09
4	Wrong Answer	2025-12-09
3	Wrong Answer	2025-12-09
2	Compile Error	2025-12-09
1	Compile Error	2025-12-09

状态: **Accepted**

源代码

```
n, m = list(map(int, input().split()))
min_circ = float('inf')
birds = []
asq = []
for _ in range(n):
    i, j = list(map(int, input().split()))
    birds.append(i)
    if i + j < min_circ:
        min_circ = i + j
        asq.append(max(i, j))

early_birds = []
late_birds = [asq[-1]]
for i in birds:
    if i <= min_circ // 2:
        early_birds.append(i)
    elif min_circ // 2 < i <= asq[-1]:
        late_birds.append(i)
early_birds.sort()
late_birds.sort()

cnt = 0
sgm = 0
s0 = sum(early_birds)
l0 = len(early_birds)
if m == s0:
    print(l0)
elif m < s0:
    init_eb = 0
    while sgm <= m - early_birds[init_eb]:
        sgm += early_birds[init_eb]
        init_eb += 1
    print(init_eb)
else:
    m -= s0
    cnt += l0
    cnt += 2 * m // min_circ
    m = m % min_circ
    print(cnt)
```

基本信息

#: 51201161

题目: 30204

提交人:

25n2500011906(Little(25n2500011906))

内存: 36412kB

时间: 233ms

语言: PyPy3

提交时间: 2025-12-09 03:32:22

P14635 [NOIP2025] 糖果店 / candy（官方数据）

提交答案

加入题单

复制题目

提交

通过

时间限制

内存限制

28.75k

7.93k

1.00s

512.00MB

题目描述

复制 Markdown

展开

进入 IDE 模式

小 X 开了一家糖果店，售卖 n 种糖果，每种糖果均有无限颗。对于不同种类的糖果，小 X 采用了不同的促销策略。具体地，对于第 i ($1 \leq i \leq n$) 种糖果，购买第一颗的价格为 x_i 元，第二颗为 y_i 元，第三颗又变回 x_i 元，第四颗则为 y_i 元，以此类推。

小 R 带了 m 元钱买糖果。小 R 不关心糖果的种类，只想得到数量尽可能多的糖果。你需要帮助小 R 求出， m 元钱能购买的糖果数量的最大值。

输入格式

输入的第一行包含两个正整数 n, m ，代表糖果的种类数和小 R 的钱数。

输入的第 $i + 1$ ($1 \leq i \leq n$) 行包含两个正整数 x_i, y_i ，分别表示购买第 i 种糖果时第奇数颗的价格和第偶数颗的价格。

输出格式

输出一行一个非负整数，表示 m 元钱能购买的糖果数量的最大值。

输入输出样例

输入 #1

复制

输出 #1

复制

2 10

4 1

3 3

4

题目编号

P14635

提供者

kkksc03

难度

普及/提高-

历史分数

85

提交记录

查看题解

题目反馈

标签

NOIP 提高组

2025

O2 优化

显示算法标签

相关讨论

进入讨论版

查看讨论

推荐题目

暂无

洛谷 / 评测记录

评测记录

查找记录

P14635

1883345

搜索

筛选选项

记录状态

全部状态

代码语言

全部语言

排序方式

最新提交

共计 2 条结果

清除所有筛选条件

LittleBeetroot

12-09 02:50:59

Unaccepted

85

P14635 [NOIP2025] 糖果店 / candy（官方数据）

2.02s / 8.01MB / 794B Python 3

LittleBeetroot

12-09 02:47:05

Unaccepted

45

P14635 [NOIP2025] 糖果店 / candy（官方数据）

2.02s / 8.02MB / 838B Python 3

源代码 [复制](#)

```
n, m = list(map(int, input().split()))
min_circ = float('inf')
birds = []
asq = []
for _ in range(n):
    i, j = list(map(int, input().split()))
    birds.append(i)
    if i + j < min_circ:
        min_circ = i + j
        asq.append(max(i, j))

early_birds = []
late_birds = [asq[-1]]
for i in birds:
    if i <= min_circ // 2:
        early_birds.append(i)
    elif min_circ // 2 < i <= asq[-1]:
        late_birds.append(i)
early_birds.sort()
late_birds.sort()

cnt = 0
sgm = 0
s0 = sum(early_birds)
l0 = len(early_birds)
if m == s0:
    print(l0)
elif m < s0:
    init_eb = 0
    while sgm <= m - early_birds[init_eb]:
        sgm += early_birds[init_eb]
        init_eb += 1
    print(init_eb)
else:
    m -= s0
    cnt += l0
    cnt += 2 * m // min_circ
    m %= min_circ
    print(cnt)
```

所属题目 [P14635 \[NOIP2025\] 糖果店 / ca...](#)

评测状态 **Unaccepted**

评测分数 **85**

提交时间 2025-12-09 02:50:59

R252250175 记录详情

编程语言 Python 3 代码长度 794B 用时 2.02s 内存 8.01MB

测试点信息 源代码

测试点信息

#1 WA 20ms/3.91MB	#2 AC 20ms/4.07MB	#3 AC 20ms/4.13MB	#4 WA 20ms/3.95MB	#5 AC 19ms/4.13MB	#6 AC 20ms/3.91MB	#7 AC 20ms/4.15MB
#8 AC 20ms/3.99MB	#9 AC 20ms/4.04MB	#10 AC 22ms/4.07MB	#11 WA 22ms/4.16MB	#12 AC 22ms/3.94MB	#13 AC 22ms/4.17MB	#14 AC 256ms/7.99MB
#15 AC 247ms/7.88MB	#16 AC 250ms/7.88MB	#17 AC 248ms/7.88MB	#18 AC 253ms/8.01MB	#19 AC 252ms/7.97MB	#20 AC 250ms/7.88MB	

测试数据下载

测试点 #1: [下载数据](#)

洛谷免费提供该记录第一个非AC的输入输出数据下载；部分题目因为版权等原因，不开放数据下载。

该功能仅限已实名认证的用户使用。每日可下载数据的次数有一定限制：灰名不可下载数据，蓝名24小时内可以下载1次，绿名2次，橙名3次，红名4次。



所属题目 P14635 [NOIP2025] 糖果店 / ca...

评测状态 Unaccepted

评测分数 85

提交时间 2025-12-09 02:50:59

2. 学习总结和收获

通过这次月考，我更加深刻地认识到自己的不足：我对于栈、队列等数据类型掌握地不够熟练(E29946)，不了解状态压缩算法(T30201)，对二进制和位运算不能灵活使用(T30201)，哈希表和矩阵综合应用会犯糊涂(M27371)，不能短时间内灵活处理较难的贪心问题(T30204)。

接下来的几周还需要在这些方面多多加强。

如果作业题目简单，有否额外练习题目，比如：OJ“计概2025fall每日选做”、CF、LeetCode、洛谷等网站题目。

这周较忙，仅在CodeForces上参加周赛并少量刷题。

LittleBeetroot submissions

#	When	Who	Problem	Lang	Verdict	Time	Memory
352594859	Dec/09/2025 01:25 ^{UTC+8}	LittleBeetroot	C - Kanade's Perfect Multiples	PyPy 3-64	Time limit exceeded on test 15	2000 ms	31700 KB
352594501	Dec/09/2025 01:23 ^{UTC+8}	LittleBeetroot	C - Kanade's Perfect Multiples	Rust 2024	Accepted	46 ms	19100 KB
352118212	Dec/06/2025 02:57 ^{UTC+8}	LittleBeetroot	A - Pizza Time	Rust 2024	Accepted	31 ms	0 KB
352117659	Dec/06/2025 02:53 ^{UTC+8}	LittleBeetroot	A - Pizza Time	Rust 2024	Wrong answer on test 1	0 ms	0 KB
352114161	Dec/06/2025 02:24 ^{UTC+8}	LittleBeetroot	A - Watermelon	Rust 2024	Accepted	62 ms	0 KB
352114036	Dec/06/2025 02:23 ^{UTC+8}	LittleBeetroot	A - Watermelon	Rust 2024	Accepted	124 ms	0 KB
352112172	Dec/06/2025 02:10 ^{UTC+8}	LittleBeetroot	A - Watermelon	Rust 2024	Compilation error	0 ms	0 KB
352111973	Dec/06/2025 02:09 ^{UTC+8}	LittleBeetroot	A - Watermelon	Rust 2024	Runtime error on test 1	30 ms	0 KB
352090575	Dec/06/2025 00:16 ^{UTC+8}	LittleBeetroot	F - Isla's Memory Thresholds	PyPy 3-64	Time limit exceeded on pretest 5	6000 ms	19600 KB
352088874	Dec/06/2025 00:13 ^{UTC+8}	LittleBeetroot	F - Isla's Memory Thresholds	PyPy 3-64	Runtime error on pretest 1	62 ms	2700 KB
352087819	Dec/06/2025 00:10 ^{UTC+8}	LittleBeetroot	F - Isla's Memory Thresholds	PyPy 3-64	Runtime error on pretest 1	31 ms	2700 KB
352076764	Dec/05/2025 23:47 ^{UTC+8}	LittleBeetroot	C - Kanade's Perfect Multiples	PyPy 3-64	Time limit exceeded on pretest 15	2000 ms	28800 KB
352076472	Dec/05/2025 23:47 ^{UTC+8}	LittleBeetroot	C - Kanade's Perfect Multiples	Python 3	Time limit exceeded on pretest 15	2000 ms	26100 KB
352073132	Dec/05/2025 23:40 ^{UTC+8}	LittleBeetroot	C - Kanade's Perfect Multiples	Python 3	Time limit exceeded on pretest 15	2000 ms	26200 KB
352072842	Dec/05/2025 23:40 ^{UTC+8}	LittleBeetroot	C - Kanade's Perfect Multiples	Python 3	Time limit exceeded on pretest 15	2000 ms	26200 KB
352071240	Dec/05/2025 23:37 ^{UTC+8}	LittleBeetroot	C - Kanade's Perfect Multiples	PyPy 3-64	Time limit exceeded on pretest 15	2000 ms	28400 KB
352068198	Dec/05/2025 23:32 ^{UTC+8}	LittleBeetroot	C - Kanade's Perfect Multiples	PyPy 3-64	Time limit exceeded on pretest 15	2000 ms	26300 KB
352047655	Dec/05/2025 23:01 ^{UTC+8}	LittleBeetroot	B - Niko's Tactical Cards	PyPy 3-64	Accepted	109 ms	15400 KB
352037843	Dec/05/2025 22:53 ^{UTC+8}	LittleBeetroot	B - Niko's Tactical Cards	PyPy 3-64	Wrong answer on pretest 1	62 ms	0 KB
352027995	Dec/05/2025 22:44 ^{UTC+8}	LittleBeetroot	A - Sleeping Through Classes	PyPy 3-64	Accepted	93 ms	2800 KB