實作題

第 1 題 成績指標

問題描述

一次考試中，於所有及格學生中獲取最低分數者最為幸運，反之，於所有不及格同學中，獲取最高分數者，可以說是最為不幸，而此二種分數，可以視為成績指標。

請你設計一支程式，讀入全班成績(人數不固定)，請對所有分數進行排序，並分別找出不及格中最高分數，以及及格中最低分數。

當找不到最低及格分數，表示對於本次考試而言，這是一個不幸之班級，此時請你印出：「worst case」；反之，當找不到最高不及格分數時，請你印出「best case」。
註：假設及格分數為 60，每筆測資皆為 0~100 間整數，且筆數未定。

輸入格式

第一行輸入學生人數，第二行為各學生分數(0~100 間)，分數與分數之間以一個空白間格。每一筆測資的學生人數為 1~20 的整數。

輸出格式

每筆測資輸出三行。
第一行由小而大印出所有成績，兩數字之間以一個空白間格，最後一個數字後無空白；
第二行印出最高不及格分數，如果全數及格時，於此行印出 best case；
第三行印出最低及格分數，當全數不及格時，於此行印出 worst case。

範例一：輸入

```
10
0 11 22 33 55 66 77 99 88 44
```

範例一：正確輸出

```
0 11 22 33 44 55 66 77 88 99
55
66
```

（說明）不及格分數最高為 55，及格分數最低為 66。

範例二：輸入

```
1
13
```

範例二：正確輸出

```
13
13
worst case
```

（說明）由於找不到最低及格分，因此第三行須印出「worst case」。

範例三：輸入

```
2
73 65
```

範例三：正確輸出

```
65 73
best case
65
```

（說明）由於找不到不及格分，因此第二行須印出「best case」。

評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制(time limit)均為 2 秒，依正確通過測資筆數給分。

10503P1.py - D:/APCS/APCS考古題/10503P1.py (3.8.1)

File Edit Format Run Options Window Help

```
#取得資料筆數 n
n = int(input())
#取得資料 inStrs, scores
inStrs = input().split()
scores = []
for i in range(n):
    scores.append(int(inStrs[i]))
#排序
scores.sort()
#搜尋最高不及格分數 highFail 及最低及格分數 lowPass
highFail, lowPass = 0,0
#scores[0] >= 60：highFail = 'best case', lowPass = scores[0]
if(scores[0]>=60):
    highFail = 'best case'
    lowPass = scores[0]
#scores[n-1] < 60：highFail = scores[n-1],  lowPass = 'worst case'
elif(scores[n-1]<60):
    highFail = scores[n-1]
    lowPass = 'worse case'
else:
    #i => 0 - n-2
    for i in range(n-1):
        highFail = scores[i]
        lowPass = scores[i+1]
        #lowPass >=60 中斷
        if(lowPass>=60):
            break
#列印結果
for s in scores:
    print(s, end=' ')
print()
print(highFail)
print(lowPass)
```

Ln: 9 Col: 13

範例一：正確輸出

```
0 11 22 33 44 55 66 77 88 99
55
66
```

找出最大不及格
最小及格

實作題

第 2 題 矩陣轉換

問題描述

矩陣是將一群元素整齊的排列成一個矩形，在矩陣中的橫排稱為列 (row)，直排稱為行 (column)，其中以 X_{ij} 來表示矩陣 X 中的第 i 列第 j 行的元素。如圖一中，$X_{32} = 6$。

我們可以對矩陣定義兩種操作如下：

翻轉：即第一列與最後一列交換、第二列與倒數第二列交換、...依此類推。

旋轉：將矩陣以順時針方向轉 90 度。

例如：矩陣 X 翻轉後可得到 Y，將矩陣 Y 再旋轉後可得到 Z。

X

1	4
2	5
3	6

Y

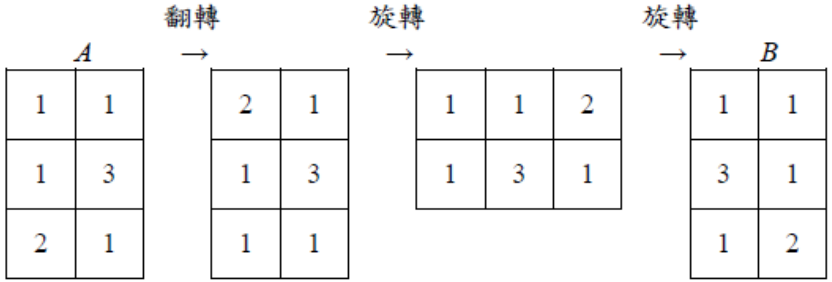
3	6
2	5
1	4

Z

1	2	3
4	5	6

圖一

一個矩陣 A 可以經過一連串的旋轉與翻轉操作後，轉換成新矩陣 B。如圖二中，A 經過翻轉與兩次旋轉後，可以得到 B。給定矩陣 B 和一連串的操作，請算出原始的矩陣 A。



圖二

輸入格式

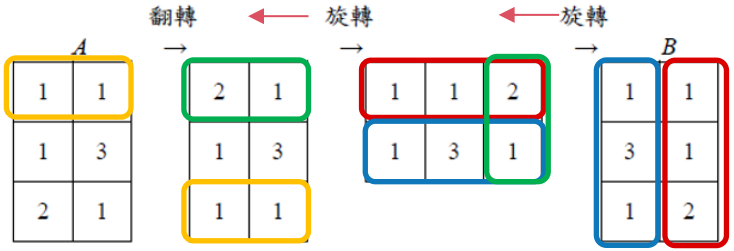
第一行有三個介於 1 與 10 之間的正整數 R, C, M。接下來有 R 行(line)是矩陣 B 的內容，每一行(line)都包含 C 個正整數，其中的第 i 行第 j 個數字代表矩陣 B_{ij} 的值。在矩陣內容後的一行有 M 個整數，表示對矩陣 A 進行的操作。第 k 個整數 m_k 代表第 k 個操作，如果 $m_k = 0$ 則代表旋轉，$m_k = 1$ 代表翻轉。同一行的數字之間都是以一個空白間格，且矩陣內容為 0~9 的整數。

輸出格式

輸出包含兩個部分。第一個部分有一行，包含兩個正整數 R' 和 C'，以一個空白隔開，分別代表矩陣 A 的列數和行數。接下來有 R' 行，每一行都包含 C' 個正整數，且每一行的整數之間以一個空白隔開，其中第 i 行的第 j 個數字代表矩陣 A_{ij} 的值。每一行的最後一個數字後並無空白。

範例一：輸入

```
3 2 3
1 1
3 1
1 2
1 0 0
```



範例一：正確輸出

```
3 2
1 1
1 3
2 1
```

（說明）

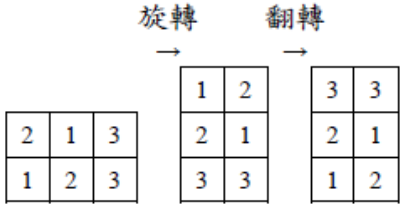
範例二：輸入

```
3 2 2
3 3
2 1
1 2
0 1
```

範例二：正確輸出

```
2 3
2 1 3
1 2 3
```

（說明）



評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制(time limit)均為 2 秒，依正確通過測資筆數給分。其中：

第一子題組共 30 分，其每個操作都是翻轉。

第二子題組共 70 分，操作有翻轉也有旋轉。

解題思路

讀取資料

R C M :列 欄 操作次數
讀取R次，切割為C筆資料，加入二維List
讀取操作，切割為 M 筆操作

操作資料

反向順序

0 -> 旋轉(rotate) / 1 -> 翻轉(flip)

輸出

輸出 R C

輸出二維List:輸出R次，每列有C筆資料

10503P2.py - D:\APCSClass3\Solutions\10503\10503P...

File Edit Format Run Options Window Help

```
def rotate(matrixB):
    matrixA = []
    r = len(matrixB)
    c = len(matrixB[0])
    for i in range(c-1, -1, -1):
        row = []
        for j in range(r):
            row.append(matrixB[j][i])
        matrixA.append(row)
    return matrixA

def flip(matrixB):
    matrixA = []
    r = len(matrixB)
    for i in range(r-1, -1, -1):
        matrixA.append(matrixB[i])
    return matrixA
```

注意方向

```
#讀取資料 R C M：列 欄 操作次數
inStrs = input().split(' ')
R = int(inStrs[0])
C = int(inStrs[1])
M = int(inStrs[2])

#讀取R次，切割為C筆資料，加入二維List
matrix = []
for i in range(R):
    temp = input().split(' ')
    rList = []
    for j in range(C):
        rList.append(int(temp[j]))
    matrix.append(rList)

#讀取操作，切割為 M 筆操作
ops = input().split(' ')
```

```
#操作資料 : 0->旋轉(rotate) / 1->翻轉(flip)
#反向順序
for k in range(M-1, -1, -1):
    op = int(ops[k])
    if(op==0):
        matrix = rotate(matrix)
        #print('rotate', matrix)
    elif(op==1):
        matrix = flip(matrix)
        #print('flip', matrix)

#輸出 R C
Rr = len(matrix)
Rc = len(matrix[0])
print(Rr, Rc)

#輸出二維List：輸出R次，每列有C筆資料
for i in range(Rr):
    for j in range(Rc):
        print(matrix[i][j], end=' ')
    print()
```

Ln: 43 Col: 9

IDLE Shell 3.9.7

File Edit Shell Debug Options Window Help

```
>>>
= RESTART: D:\APCSClass3\Solutions\10503\10503P2.py
3 2 3
1 1
3 1
1 2
1 0 0
3 2
1 1
1 3
2 1
>>>
= RESTART: D:\APCSClass3\Solutions\10503\10503P2.py
3 2 2
3 3
2 1
1 2
0 1

2 3
2 1 3
1 2 3
>>>
```

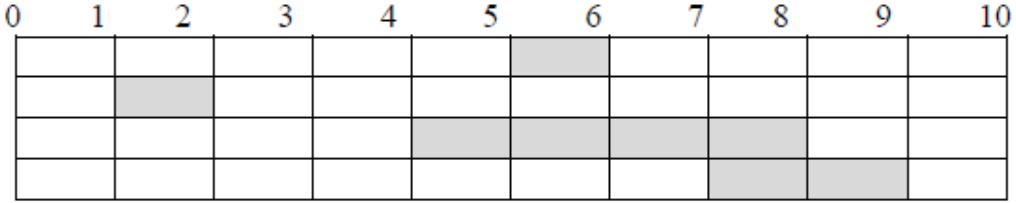
Ln: 23 Col: 4

實作題

第 3 題 線段覆蓋長度

問題描述

給定一維座標上一些線段，求這些線段所覆蓋的長度，注意，重疊的部分只能算一次。例如給定三個線段：(5, 6)、(1, 2)、(4, 8)、和(7, 9)，如下圖，線段覆蓋長度為 6。



輸入格式：

第一列是一個正整數 N，表示此測試案例有 N 個線段。

接著的N 列每一列是一個線段的開始端點座標和結束端點座標整數值，開始端點座標值小於等於結束端點座標值，兩者之間以一個空格區隔。

輸出格式：

輸出其總覆蓋的長度 。

範例一：輸入

輸入	說明
5	此測試案例有 5 個線段
160 180	開始端點座標值與結束端點座標
150 200	開始端點座標值與結束端點座標
280 300	開始端點座標值與結束端點座標
300 330	開始端點座標值與結束端點座標
190 210	開始端點座標值與結束端點座標

範例一：輸出

輸出	說明
110	測試案例的結果

160~180 可以被 150~200取代

190~210可取代 150~200 改為150~210

範例二：輸入

輸入	說明
1	此測試案例有 1 個線段
120 120	開始端點座標值與結束端點座標值

範例二：輸出

輸出	說明
0	測試案例的結果

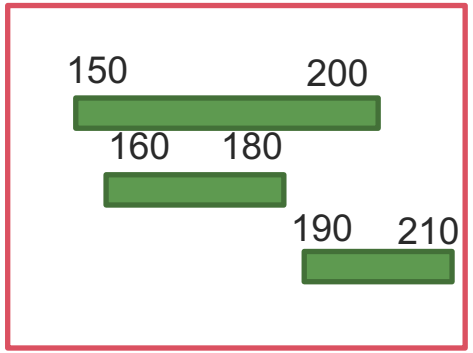
評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制(time limit)均為 2 秒，依正確通過測資筆數給分。每一個端點座標是一個介於 0~M 之間的整數，每筆測試案例線段個數上限為 N。其中：

第一子題組共 30 分，M<1000，N<100，線段沒有重疊。

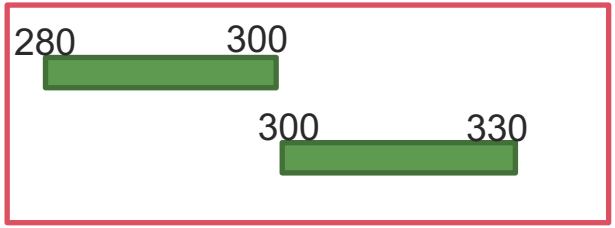
第二子題組共 40 分，M<1000，N<100，線段可能重疊。

第三子題組共 30 分，M<10000000，N<10000，線段可能重疊。



A 尾大等於 B 頭
需要合併
1 B 頭尾都小於 A
忽略 B 線段
2 B尾大於 A 尾
B 尾取代 A 尾

```
#將線段合併處理為不重疊線段，加入resultLines二維列表
#以第一組線段(lines[0])作為當前線段curline
resultLines=[]
curline = lines[0]
#將lines[i] 由索引1-n 逐一取出
for i in range(1, n):
    #若當前線段與lines[i]沒有重疊，當前線段加入resultLines
    if(curline[1]<lines[i][0]):
        resultLines.append(curline)
        curline = lines[i]
    #若當前線段與lines[i] 重疊且curline終點<lines[i]終點，合併線段
    elif(curline[1]<lines[i][1]):
        curline = [curline[0], lines[i][1]]
#將最後的當前線段加入resultLines
resultLines.append(curline)
#print(resultLines)
```



60 + 50 = 110

150	200
160	180
190	210
280	300
300	330

10503P3.py - D:\APCS\APCS考古題\10503P3.py (3.8.1)

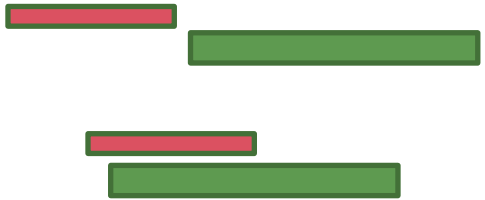
File Edit Format Run Options Window Help

```
#讀取線段數量 n
n = int(input())

#讀取n次字串，切割組成線段起點及終點列表，加入 lines 二維列表
lines=[]
for i in range(n):
    pairs = input().split(' ')
    lines.append([int(pairs[0]), int(pairs[1])])

#將 lines 以線段起點排序
lines.sort(key=lambda x:x[0])
#print(lines)
```

```
#將線段合併處理為不重疊線段，加入resultLines二維列表
#以第一組線段(lines[0])作為當前線段curline
resultLines=[]
curline = lines[0]
#將lines[i] 由索引1-n 逐一取出
for i in range(1, n):
    #若當前線段與lines[i]沒有重疊，當前線段加入resultLines
    if(curline[1]<lines[i][0]):
        resultLines.append(curline)
        curline = lines[i]
    #若當前線段與lines[i] 重疊且curline終點<lines[i]終點，合併線段
    elif(curline[1]<lines[i][1]):
        curline = [curline[0], lines[i][1]]
#將最後的當前線段加入resultLines
resultLines.append(curline)
#print(resultLines)
```

```
#輸出不重疊線段總長度
length = 0
for rLine in resultLines:
    length = length + (rLine[1]-rLine[0])

print(length)
```

因為排序所以現在起點一定小於下一個起點



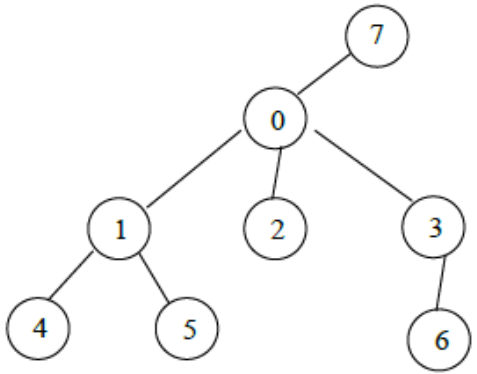
實作題

第 4 題 血緣關係

問題描述

小宇有一個大家族。有一天，他發現記錄整個家族成員和成員間血緣關係的家族族譜。小宇對於最遠的血緣關係 (我們稱之為"血緣距離") 有多遠感到很好奇。

右圖為家族的關係圖。0 是 7 的孩子，1、2 和 3 是 0 的孩子，4 和 5 是 1 的孩子，6 是 3 的孩子。我們可以輕易的發現最遠的親戚關係為 4(或 5)和 6，他們的"血緣距離"是 4 (4~1，1~0，0~3，3~6)。



給予任一家族的關係圖，請找出最遠的"血緣距離"。你可以假設只有一個人是整個家族成員的祖先，而且沒有兩個成員有同樣的小孩。

評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制(time limit)均為 3 秒，依正確通過測資筆數給分。其中，

第 1 子題組共 10 分，整個家族的祖先最多 2 個小孩，其他成員最多一個小孩，$2 \le n \le 100$ 。

第 2 子題組共 30 分，$2 \le n \le 100$。

第 3 子題組共 30 分，$101 \le n \le 2{,}000$。

第 4 子題組共 30 分，$1{,}001 \le n \le 100{,}000$。

輸入格式

第一行為一個正整數 n 代表成員的個數，每人以 0~n-1 之間惟一的編號代表。接著的 n-1 行，每行有兩個以一個空白隔開的整數 a 與 b (0 ≤ a,b ≤ n-1)，代表 b 是 a 的孩子。

B是A的孩子

輸出格式

每筆測資輸出一行最遠"血緣距離"的答案。

範例一：輸入

```
8
0 1
0 2
0 3
7 0
1 4
1 5
3 6
```

範例一：正確輸出

```
4
```

（說明）

如題目所附之圖，最遠路徑為 4->1->0->3->6 或 5->1->0->3->6，距離為 4。

範例二：輸入

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

範例二：正確輸出

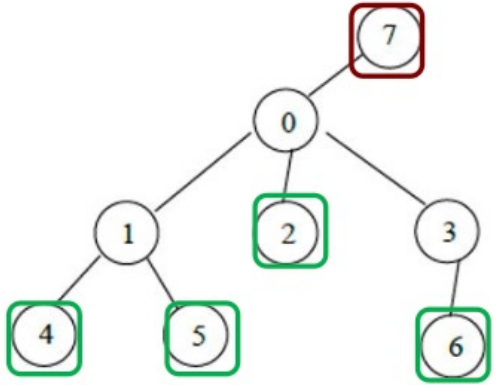
```
3
```

（說明）

最遠路徑為 1->0->2->3，距離為 3。

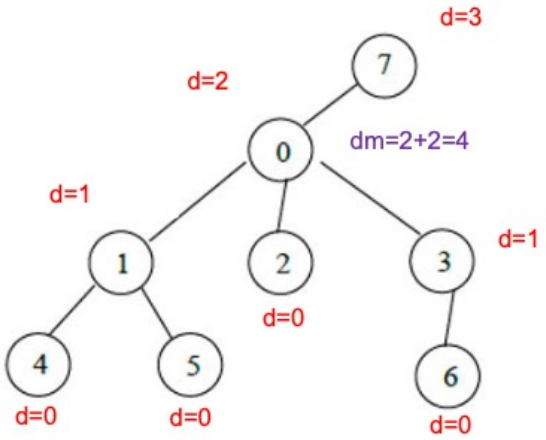
解題思路

- 取得輸入建立血緣關係圖形
  - 以dict儲存父節點對應之子節點，子節點以list儲存
- 最遠血緣關係發生位置
  - 根節點與葉節點
  - 葉節點與葉節點
- 分析根節點及葉節點
  - 根節點：節點不為其他節點之子節點
  - 葉節點：節點沒有相鄰節點



解題思路

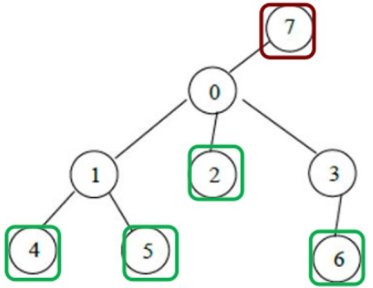
- 以dfs從根節點開始找尋深度
  - 葉節點傳回0
  - 一個子節點，深度為子節點往下深度+1
  - 一個以上子節點，取得葉節點中最大深度
- 最遠血緣關係
  - 根節點起算到葉節點的深度
  - 節點到兩個不同葉節點深度相加
  - 兩者取大的為最遠血緣關係

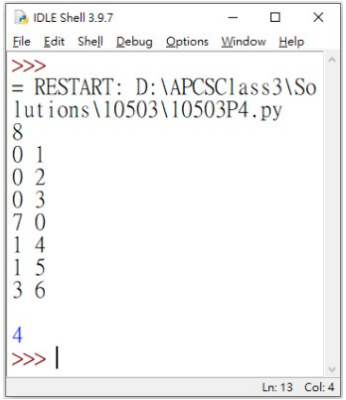


```
N = int(input())
child = {}                                  #血緣圖形
for i in range(N):                          #字典儲存子節點資訊
    child[i] = []
inStr = input()
while inStr:
    a,b = map(int, inStr.split())
    child[a].append(b)                      #儲存各節點對應子節點資訊
    inStr = input()

root = list(range(N))
for i in child:
    for j in child[i]:
        root.remove(j)                      #節點不為其他節點之子節點為根節點

blood_distance = 0                          #最遠血緣關係
from_root = dfsDistance(root[0])            #由根節點計算葉節點最大深度
blood_distance = max(from_root, blood_distance)  #與節點到兩個不同葉節點深度相加比較大小
print(blood_distance)
```

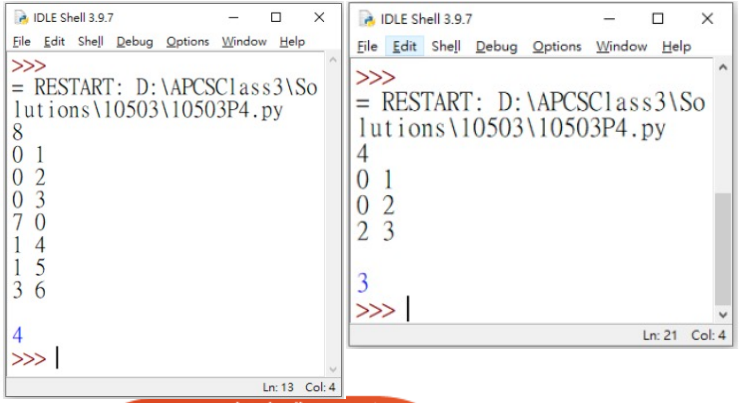
輸入將父節點與子節點並建立對應關係

將root內的節點移除 可得到root節點





```python
def dfsDistance(x):                                  #計算中間過程最大深度
    global child, blood_distance
    if len(child[x]) == 0:                           #沒有小孩是遞迴的出口條件
        return 0
    elif len(child[x])==1:                           #只有一個子節點
        return dfsDistance(child[x][0])+1
    else:                                            #2個子節點以上
        max1, max2 = 0, 0                            #走訪每個子節點，找出最大深度前兩名 max1, max2
        for ch in child[x]:                          #逐一處理每一個子節點
            depth = dfsDistance(ch) + 1              #取得此子節點最大深度
            if depth > max1:                         #此子節點
                max1, depth = depth, max1
            if depth > max2:
                max2 = depth
            #print(f"max1:{max1} max2:{max2}")
        blood_distance = max(blood_distance, max1+max2)
        return max1
```




```
max1:1 max2:0
max1:1 max2:1
max1:2 max2:0
max1:2 max2:1
max1:2 max2:2
```