程式作業五

107502540陳皇宇

**資料結構**：

class Vertex

selfloop = 0

edge = []

class Edge

v1 is first vertex

v2 is second vertex

visied = False

**演算法**：

先計算所有vertex 的edge是不是偶數(如果不是偶數一定not exist)

再把每個vertex的edge按大小sort

接著跑DFS 每當DFS return 就把點加入AnsPath

跑完後判斷edge的數量是否與AnsPath相等

最後把AnsPath倒著print

**時間複雜度**：

由於我的寫法每個edge只會進去一次(只要走過就把edge remove)且只會挑一個點進入DFS，所以時間複雜度為 O(E)，但還要加讓前面sort edge的時間O(ElogE)

**Time complexity O(E+ElogE)=O(E)**

**Pseudo code**：

function DFSbyWhile()

stack = []

stack.append([first vertex,first edge of first vertex])

back = False

while len(stack) != 0:

recur = stack[-1]//recur[0]為 vertex，recur[1]為剛剛進入的edge的vertex

if back

Anspath.append(recur[1])

ve = recur[0].edge

bre = False

back = False

if ve != []//永遠只跑第一個edge因為事先排序過且跑過的會del

stack[-1][1] = ve[0]//即將進入下一個點因此把vertex記錄起來等等return就可以把這個vertex加入AnsPath

stack.append([ve[0], 0])

recur[0].edge.remove(recur[1])

recur[1].edge.remove(recur[0])

bre = True

if bre

continue

stack.pop()

back = True