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
```