CODIGO EN PYTHON

```
import sys
import threading
import time
start_time = time.time()

def leerGraph(filename):
    f = open(filename)

    adyacenciaList = []
    adyacenciaList_reversed = []

    line = f.readline()
    while line != '':
        num1, num2 = line.split()
        v_from = int(num1)
        v_to = int(num2)
        max_v = max(v_from, v_to)

        while len(adyacenciaList) < max_v:
            adyacenciaList.append([])

        while len(adyacenciaList_reversed) < max_v:
            adyacenciaList_reversed.append([])

        adyacenciaList[v_from-1].append(v_to-1)
        adyacenciaList_reversed[v_to-1].append(v_from-1)

        line = f.readline()

    return adyacenciaList, adyacenciaList_reversed

def Deep_first_search_Loop1(graph_rev, n):

    global t, visitado, orden_por_horaLlegada

    t = 0
    visitado = [False]*n
    orden_por_horaLlegada = [None]*n

    for i in reversed(range(n)):
        if not visitado[i]:
            DFS_1(graph_rev, i)
```

```
def DFS_1(graph_rev, i):

    global t, visitado

    visitado[i] = True

    for v in graph_rev[i]:
        if not visitado[v]:
            DFS_1(graph_rev, v)

    orden_por_horaLlegada[t] = i
    t += 1


def Deep_first_search_Loop2(graph):

    global scc_tam, visitado, orden_por_horaLlegada

    visitado = [False]*len(graph)
    res = []

    for i in reversed(range(len(graph))):
        if not visitado[orden_por_horaLlegada[i]]:
            scc_tam = 0
            DFS_2(graph, orden_por_horaLlegada[i])
            res.append(scc_tam)

    return res


def DFS_2(graph, i):

    global visitado, scc_tam

    visitado[i] = True

    for v in graph[i]:
        if not visitado[v]:
            DFS_2(graph, v)

    scc_tam += 1


def SCCkosaraju(graph, graph_rev):

    Deep_first_search_Loop1(graph_rev, len(graph))
    res = Deep_first_search_Loop2(graph)

    return res
```

```
t = 0
s = None
visitado = None
leader = None
scc_tam = 0
orden_por_horaLlegada = None

def main():

    graph, graph_rev = leerGraph('SCC.txt')

    res = SCCkosaraju(graph, graph_rev)

    print('--- Resultados ---')
    print(','.join(map(lambda x: str(x), sorted(res)[::-1][:5])))
    print("--- %s seconds ---" % (time.time() - start_time))

if __name__ == '__main__':
    threading.stack_size(67108864)
    sys.setrecursionlimit(2 ** 20)
    thread = threading.Thread(target = main)
    thread.start()
```