



Código Asignatura:
ISC-314

Nombre:
Ronald Mariotti

Matricula:
2014-0698

Trabajo:
Tarea 8 - Componentes Fuertemente Conectados

README

Esta tarea lo que se consiste es en codificar y ejecutar el algoritmo de Kosaraju para el cálculo de componentes fuertemente conectados (SCC). Lo que se busca es indicar el tamaño de los 5 SCC que me se me da. Si el algoritmo calcula bien los 5 tamaños de los SCC mas grandes que fuesen 500, 400, 300, 200 y 100, su respuesta debería ser "500.400.300.200.100". Por lo tanto, si el algoritmo calcula sólo 3 SCC cuyos tamaños son de 400, 300 y 100, su respuesta debería ser "400,300,100,0,0".

- ADVERTENCIA: Esta es la tarea más difícil de la programación del curso. Debido al tamaño del grafo puede tener que administrar la memoria con cuidado. Especialmente porque se usaran algoritmos recursivos cuando hagas el recorrido DFS (Deep First Search).
- En el reporte me justificaras la técnica utilizada.

La técnica de el algoritmo de Kosaraju consiste en usar dos DFS. La primera, en el grafo original, se utiliza para elegir el orden en que el loop de la segunda DFS comprueba los vértices que han sido visitados ya y explora recursivamente si no, y cada exploración recursiva encuentra una sola nueva componente fuertemente conectada.

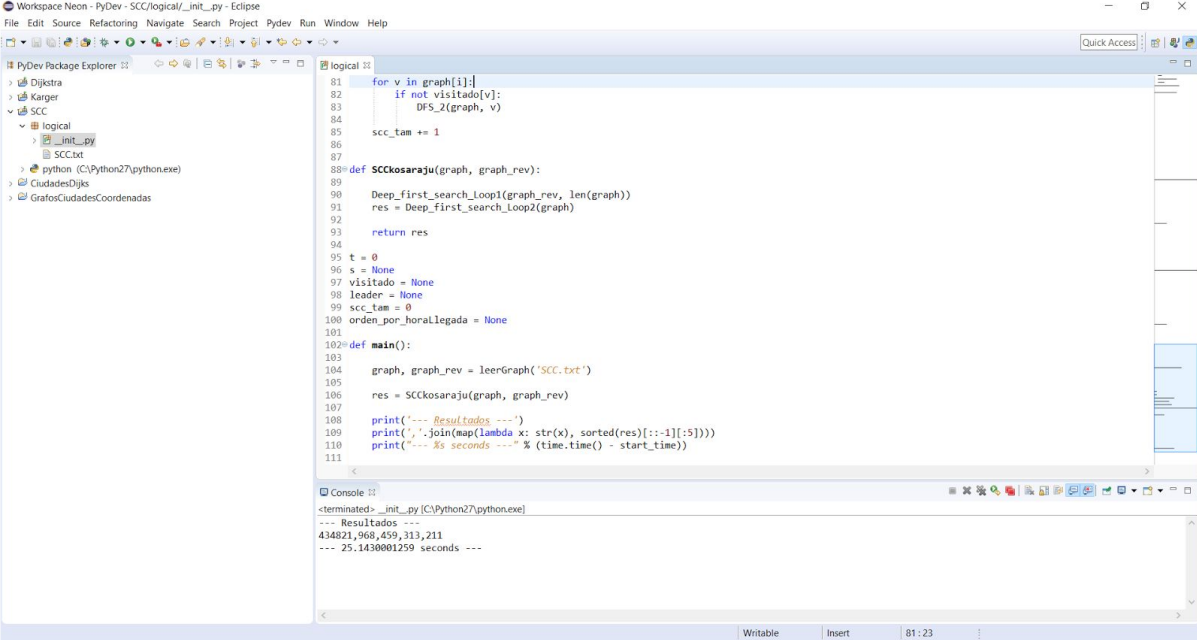
Resultados del programa

Tamaño de los 5 mayores elementos fuertemente conectados:

434821 , 968 , 459 , 313 , 211

Tiempo = 25.1430001259 seconds

Captura de resultados



The screenshot shows the Eclipse IDE interface with a workspace named "Workspace Neon". The project "SCC" is open, and the file "logical/_init_.py" is being edited. The code implements a function `SCCkosaraju` that uses Kosaraju's algorithm to find Strongly Connected Components (SCC) in a directed graph. The graph is loaded from a file named "SCC.txt". The results are printed to the console, showing the SCCs and the execution time.

```
81     for v in graph[i]:
82         if not visitado[v]:
83             DFS_2(graph, v)
84
85     scc_tam += 1
86
87
88 def SCCkosaraju(graph, graph_rev):
89
90     Deep_first_search_loop1(graph_rev, len(graph))
91     res = Deep_first_search_loop2(graph)
92
93     return res
94
95 t = 0
96 s = None
97 visitado = None
98 leader = None
99 scc_tam = 0
100 orden_por_hora_llegada = None
101
102 def main():
103
104     graph, graph_rev = leerGraph('SCC.txt')
105
106     res = SCCkosaraju(graph, graph_rev)
107
108     print("--- Resultados ---")
109     print(','.join(map(lambda x: str(x), sorted(res)[::-1][:5])))
110     print("--- %s seconds ---" % (time.time() - start_time))
111
```

The console output shows the results of the execution:

```
<terminated> _init_.py [C:\Python27\python.exe]
--- Resultados ---
434821,968,459,313,211
--- 25.1430801259 seconds ---
```