

Laboratorio Nro. 6: Implementación de Grafos

Alejandro Murillo González

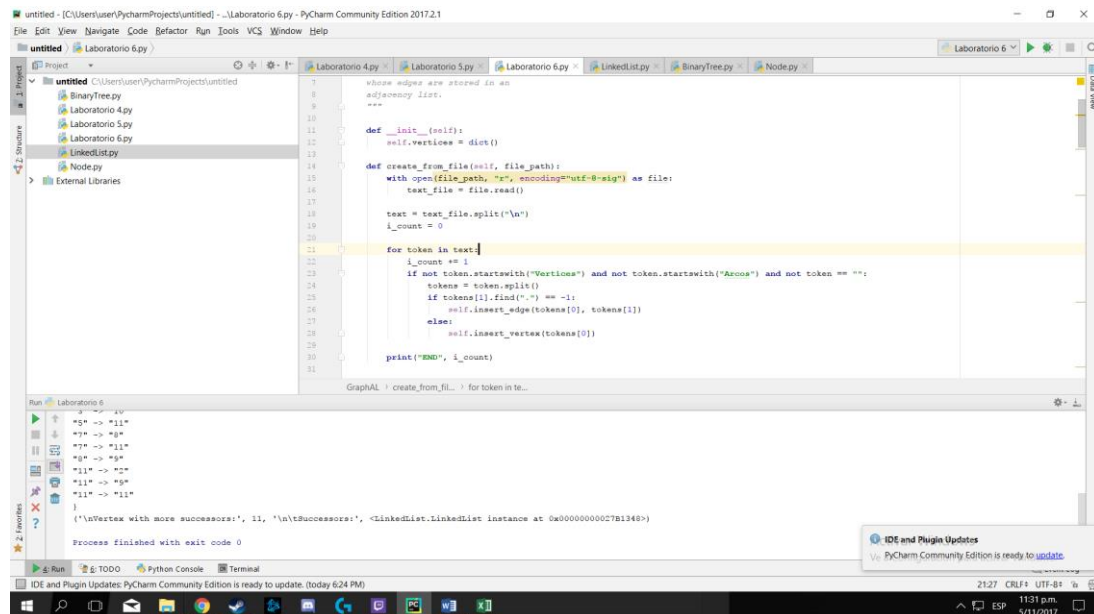
Universidad Eafit
Medellín, Colombia
amurillo@eafit.edu.co

Juan Pablo Vidal Correa

Universidad Eafit
Medellín, Colombia
Jpvidalc@eafit.edu.co

3) 3) Simulacro de preguntas de sustentación de Proyectos

3.1 Incluyan una imagen de la respuesta de las pruebas del numeral 1.3



```
untitled - [C:\Users\user\PycharmProjects\untitled] - Laboratorio 6.py - PyCharm Community Edition 2017.2.1
File Edit View Navigate Code Refactor Run Tools VCS Window Help

untitled Laboratorio 6.py
Project:
  untitled
  BinaryTree.py
  Laboratorio 4.py
  Laboratorio 5.py
  Laboratorio 6.py
  LinkedList.py
  Node.py
  External Libraries
  2 Favorites

untitled Laboratorio 6.py
8   whose edges are stored in an
9   adjacency list:
10  ---
11
12  def __init__(self):
13      self.vertices = dict()
14
15  def create_from_file(self, file_path):
16      with open(file_path, "r", encoding="utf-8-sig") as file:
17          text_file = file.read()
18
19          text = text_file.split("\n")
20          i_count = 0
21
22          for token in text:
23              i_count += 1
24              if not token.startswith("Vertices") and not token.startswith("Edges") and not token == "":
25                  tokens = token.split()
26                  if tokens[0].find(".") == -1:
27                      self.insert_edge(tokens[0], tokens[1])
28                  else:
29                      self.insert_vertex(tokens[0])
30
31          print("END", i_count)
32
33  GraphML -> create_from_fil... -> for token in te...
34
35 Run: Laboratorio 6
36 *5* -> *11*
37 *5* -> *5*
38 *7* -> *11*
39 *8* -> *5*
40 *11* -> *2*
41 *11* -> *5*
42 *11* -> *11*
43
44 ("Vertex with more successors", 11, "\n\nSuccessors", <LinkedList.LinkedList instance at 0a000000000781348>)
45
46 Process finished with exit code 0
47
48 IDE and Plugin Updates: PyCharm Community Edition is ready to update. (today 6:24 PM)
49
50 21:27 CRLF+ UTF-8+
51 11:31 p.m.
52 5/11/2017
```

3.2 Escriban una explicación entre 3 y 6 líneas de texto del código del numeral 1.1. Digan cómo funciona, cómo está implementado el grafo con matrices y con listas que hizo, destacando las estructuras de datos y algoritmos usados

Para la implementación de un grafo cuyos vértices están contenidos en una matriz de adyacencia se inserta un vértice y se verifica si no está repetido en

el grafo, después se insertan los vértices en el grafo y a cada uno de estos se le añade una arista, y a esta conexión se le asigna un peso. Después se verifica si el vértice tiene un sucesor, si no tiene este se agrega primero (el sucesor es el vértice adyacente. Por último se retornan el vertice con más sucesores.

3.3 ¿En qué grafos es más conveniente utilizar la implementación con matrices de adyacencia y en qué casos en más convenientes listas de adyacencia? ¿Por qué?

Depende el tamaño de la información que se va a guardar en cada grafo, si todos los valores pueden contenerse en una única columna de una matriz, no vale la pena emplear una matriz de adyacencia por qué se va a gastar recurso incensario, ya que la matriz tendría mucho espacio sin aprovechar, lo que para un grupo de datos pequeños significa mayor consumo de memoria y tiempo de acceso (complejidad), lo cual es contraproducente e incensario para un conjunto pequeño de datos. Lo contrario sucede con una lista de adyacencia, la cual es muy útil para conjuntos de datos pequeños, pero no es rentable para un gran volumen de datos, por que tomaría mucho tiempo en recorrer archivo por archivo hasta llegar al deseado. Por lo tanto depende del tamaño de los datos a guardar en los grafos, si es un conjunto grande se emplean matrices, de lo contrario se emplea una lista.

3.4 Para representar el mapa de la ciudad de Medellín del ejercicio del numeral 1.3, ¿Qué es mejor usar, Matrices de Adyacencia o Listas de Adyacencia? ¿Por qué?

Listas de adyacencia, porque las relaciones entre cada nodo, es decir cada calle con intercesiones se puede representar de manera eficiente en una lista, porque el tiempo de acceso es muy reducido, ya que cada relación es precisa y se puede hacer un recorrido de manera rápida.

3.5 Para una red social como Facebook, ¿Qué es mejor usar, Matrices de Adyacencia o Listas de Adyacencia? ¿Por qué?

Matrices de adyacencia, porque la cantidad de datos es muy grande, por lo que el espacio de la matriz se va a ocupar de manera eficiente, además las relaciones, que serían las amistades, se pueden representar de manera óptima en una matriz, porque en cada nodo hay múltiples relaciones. Lo que

permite un tiempo de acceso rápido, y un consumo equiparable a la cantidad manejada.

3.6 Para representar la tabla de enrutamiento, respondan: ¿Qué es mejor usar, Matrices de Adyacencia o Listas de Adyacencia?

Para una tabla de enrutamiento, el cual es un documento electrónico que almacena las rutas a los diferentes nodos en una red informática, es más eficiente utilizar listas de adyacencia, porque entre la fuente y el destino hay nodos intermedios, los cuales se pueden representar eficientemente en una lista de adyacencia.

3.7 Calculen la complejidad de los ejercicios en línea, numerales 2.1

La complejidad del ejercicio es $O(nm)$, ya que se utilizan ciclos anidados.

3.8 Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) el cálculo de complejidad del numeral 3.7

Para el ejercicio del numeral 2.1 n serían los vértices y m sus sucesores.

4) Simulacro de Parcial

1.

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

2. Recorrido DFS:

0: 3742156

1: 0372465

2: 1037546
3: 7
4: 2103756
6: 2103754

Recorrido BFS:

0: 3472165
1: 0253467
2: 1460537
3: 7
4: 2160537
6: 2140537

3. a) $O(n)$

4. Metodo:

```
children = graph.getSucesors (p)
for child in children:
    if not seen[child]:
        seen[child] = True
        myList.append(child)
    return aPathAux(graph,child,q,myList,seen);
return
```

5) Lectura recomendada (opcional)

a) Resumen:

Grafos:

Un grafo es un tipo abstracto de datos, que consiste en un conjunto de nodos (también llamados vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos.

Esta estructura permite representar problemas de la vida real, por ejemplo un nodo puede ser una ciudad y una arista los vuelos que llegan a esta, lo anterior permite evidenciar las relaciones en situaciones de la vida real.

Adyacencia: Se dice que 2 vértices son adyacentes uno del otro, si están conectados por la misma arista.

Recorrido: Es la secuencia de aristas.

Grafo conectado: Un grafo está conectado si para cada vértice hay una un recorrido que lo relacione con otro vértice.

Añadir peso a un nodo: En algunos casos se le puede dar un peso a una arista, lo cual indica la distancia física entre dos vértices o su tiempo de acceso.

6) Trabajo en Equipo y Progreso Gradual (Opcional)

a) Actas de reunión:

Integrante	Fecha	Hecho	Haciendo	Por Hacer
Murillo	1/11/2017	Implemento GraphAL en python	Implementación GraphAM	
Vidal	3/11/2017	Implemento código en línea (2.1)	Calculando complejidad	leer lectura opcional y resumen
Murillo	3/11/2017	Pruebas numeral 1.3	Organizar texto- informe de laboratorio	Agregar partes opcionales de Vidal
Vidal	5/11/2017	Implemento resumen	Simulacro de parcial	Organizar texto- Informe de laboratorio



UNIVERSIDAD EAFIT
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS

Código: ST245

Estructura de
Datos 1

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co