Laboratory practice No. 5: Graph implementation

Juan Sebastián Pérez Salazar
Universidad Eafit
Medellín, Colombia
jsperezs@eafit.edu.co

Yhoan Alejandro Guzmán García
Universidad Eafit
Medellín, Colombia
yaguzmang@eafit.edu.co

3) Practice for final project defense presentation

1. For the case of adjacency matrices, a matrix is created and a constructor that receives a size is generated. For the case of the list, a size is received and an empty LinkedList ArrayList is generated. In the getWeight method of the matrix, the position of the matrix is returned with the parameters that are received. For the lists the LinkedList of the position "source" is traversed and "destination" is searched, then the value of that object is returned. For the addArc method, the value is added to the ["source", "destination"] position and, for the lists, in the ArrayList in the "source" position, an object with "destination" and the value is added to the LinkedList. Finally, for the getSuccessors method, the row "source" is traversed and the values are added to a new ArrayList, which will be returned. For the lists, the LinkedList of the "source" position is traversed.

2. To represent the map of the city of Medellin it is better to use adjacency lists since these in comparison to adjacency matrices consume less memory, which is on average O(n * m), where n is the number of nodes and m number of relationships Although for some cases the matrices can be faster to access, in general, their average time is almost the same. Therefore, the most convenient to represent a large number of nodes is an adjacency list, as it is more memory-saving and equally effective on matrices.

3. As mentioned in the previous case with the map of the Medellin city, when we come to representing a large amount of data it is better to use an adjacency list. This is because a correct data structure is not only based on efficiency, it is also based on the memory consumption of the algorithm. Therefore, storing 100 million users in an adjacency matrix would be a complexity of $O(n^2)$ while in a list it would be O (n + m), where n would be the number of users and m the number of friends, in others words, the number of relationships with other people.

4. For this case of routing tables the best structure to use is an adjacency matrix, because these matrices serve with lower amounts of data at a higher speed than the adjacency lists. Since, in the matrices, access the values, obtain the successors or find the arcs is much faster than in the lists, since they have to make a tour with a cycle for the values it contains, to know if it is linked one vertice with another. In this specific case, as you need to know what is the shortest distance from one device to another in the network, the tables can perform this task more effectively than a list.

4) Practice for midterms

1. **Table**

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

2. 0 -> [3,4]
1 -> [0,2,5]
2 -> [1,4,6]
3 -> [7]
4 -> [2]
5 ->
6 -> [2]
7 ->

3. B) O(n^2)

5) Recommended reading (optional)

a) Title
b) Main ideas
c) Concept map

6) Team work and gradual progress (optional)

a) Meeting minutes

PROFESSOR MAURICIO TORO BERMÚDEZ
Phone: (+57) (4) 261 95 00 Ext. 9473. Office: 19 - 627
E-mail: mtorobe@eafit.edu.co

Member	Date	Done	Doing	To do
Sebastián	18/10/2018			point 1.1
Sebastián	18/10/2018	**Point 1.1 with Adyacency matrix**	point 1.1 with Adyacency lists	test for point 1.1
Sebastián	19/10/2018	**point 1.1 with Adyacency lists**	test for point 1.1	Practice for final project defense presentation
Sebastián	19/10/2018	**test for point 1.1**	Analisys for point 1.2	Practice for final project defense presentation
Sebastián	20/10/2018	**Practice for final project defense presentation**		Practice for midterms
Sebastián	21/10/2018	**Practice for midterms**		Point 1.2
Sebastián	21/10/2018	**Point 1.2**		Point 2.1
Yhoan	22/10/2018	**Point 2.1**		recommended reading
Yhoan	22/10/2018	**recommended reading**		upload the laboratory

b) History of changes of the code

History changes of code		
Version	Code	Status
1.0	1.1	
2.0	1.1	
1.0	1.2	
1.0	2.1	
2.0	2.1	