Laboratory practice No. 5: Graph Implementation

Juliana Lalinde Velásquez
Universidad Eafit
Medellín, Colombia
jlalindev@eafit.edu.co

Isabel Urrego Gómez
Universidad Eafit
Medellín, Colombia
iurregog@eafit.edu.co

3) Practice for final project defense presentation

1. In this exercise we tried two implementations of a graph, matrix and list. The first is a matrix of n x n where n is the number of vertices and each position ij has the distance between the vertex i to the vertex j. This means that to look for the weight between two vertices you just have to access to the position of the matrix where i is the source and j the destination.
   The list saves the same information as the matrix but differently. For each vertex you have a linked list of pairs, the related vertex and its weight. To know the weight between two vertices you just have to look in the position of the list that is the same as the number of source node and transverse the list to find the destination node and the corresponding weight.

2. In this case the better option would be to use adjacency lists over adjacency matrices. The reason lies on the memory required. If we used adjacency matrices we would have a space on the memory for each pair of streets even if they are not connected in real life, whereas the adjacency list saves only the information of the streets related and because of that there is no wasted memory in this case.

3. To determinate whether is better to use one or the other depends on the problem because each one has its advantages and disadvantages over the other. For example in case of the memory the adjacency list uses in the most cases less memory than the matrix because it depends on the number of edges while the matrix uses always O(n^2) even if some vertices are not related, but if we have for example a graph where all vertices are related with the others both data structures would spend the same memory and in this case there would be a tie. However, if the algorithm requires a lot to look for a presence of a specific edge, the time spent to do this function is less in the matrix than in the list. Therefore, both options should be known because in some problems one can win over the other and vice versa.

4. In this case the waste of memory is not a factor to consider because both data structures would spend the same space since each vertex in the network is related to all others. The problem to face is finding the best access route and therefore we need the fastest data structure for this function, that is why this is a problem where the adjacency matrix wins over the list if we want to minimize time. To find the weight between two nodes is O(1) in the matrix and O(n) in the list in the worst case.
5. **Complexity**

Internal loop

$$T(n) = \sum_{x=0}^{n} c_1$$

$$T(n) = \sum_{x=0}^{n} c_2 + n$$

O(n^2)

6. O(n^2), where n is the number of nodes in the graph.

4) Practice for midterms

1. *Matrix*

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

3. A. O(n^2)

5) Recommended reading (optional)

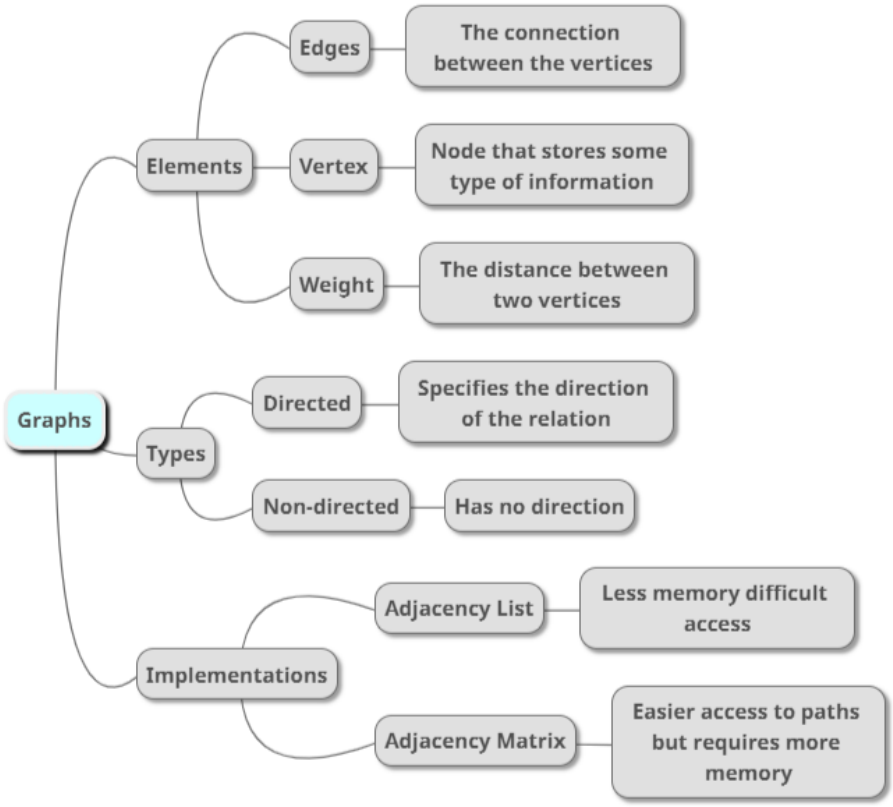
a) Chapter 13: Graphs

b) A graph is a data structure that allows to store information and know how the elements are related. It has been often related to the trees but they both have different applications since the hierarchy in a graph is not an important element whereas in the tree leads to the main structure.

This data structure has tree important elements, the vertices, the edges that show the connection between the nodes and the weight that is the distance between two vertices. In some cases the weight disappears because the information needed is whether the points are related or not and there are also two types of graphs, the directed and the non-directed. The first one indicates the direction of the

relation between the vertices because in some cases A might be related to B but there is no path that goes from B to A. On the other hand, the non-directed doesn't specify the direction of the relation.

c) Concept Map: Graphs



6) Team work and gradual progress (optional)

a) Meeting minutes

Date	Time	Description
20.10.18	1 hour	Distribution of the work
21.10.18	1 hour (by phone)	Discussion of parts 3.4 and 4
Additional work		Watching videos about space complexity

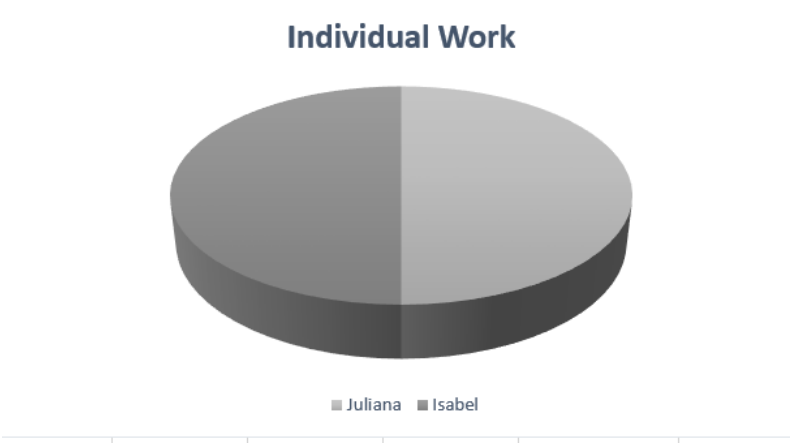
b) History of changes of the code

Version	Includes	Left	Status
1.0	1.a and b	1.2 and 2.1	
2.0	1	2.1	
3.0	1 and 2		Complete

c) History of changes of the report

Version	Includes	Left	Status
1.0	3.1,3.2,3.3,3.4 4 5	3 and rest of 4	
2.0	Part 3 Part 4 Part 5		

d) Individual work



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

Member	Part of the Laboratory	Description
Juliana	1	a
	2	2.1
	3	3.5 3.6
	4	4
	6 and 7	Team work and english
Isabel	1	b and 1.2
	2	
	3	3.1 3.2 3.3 3.4
	4	4
	5	Lecture
	6 and 7	Team work and english

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