Course: Introduction to Graph Algorithms

Professor: C Pandu Rangan

Department: Computer Science and Engineering

Institute: IISc

Week: 01

Lecture 03 Shortest Path Algorithms

Namaskara, in this session, we are going to take a look at the basics of graphs, since the focus of this series of lectures is on graph algorithms. I would like to introduce the concept of graphs and explain certain basic definitions. We will also see how the graphs can be represented in a computer. After that we get directly into the discussions related to graph algorithms. Introduction to graphs. Graphs are an interesting and important class of discrete structures.

Graphs are used to model several real-world problems in various contexts. Consider for example, the communication network and the road network or any transportation network and you have to identify a path or a route, to go from a source to a destination. Consider the social network, A and B are related when they know each other and when they are friends and so on. So, also a distributed system.

So if you consider the real world problems that are arising in context such as communication networks, social network, transportation network, problems arising in distributed systems and so on, these real world problems are naturally modeled as graphs and problems on graphs, that is the reason why this is an important area of study in design and analysis of algorithms. We have 2 kinds of graphs: directed and undirected graphs. First we will look at directed graph. A directed graph G consists of two sets. V and E, a typical notation used for these 2 sets.

$$G = (V, E)$$

V is a finite, non-empty set. E is a subset of V x V,

$$E \subseteq V \times V$$

so what is V x V, V x V is set of all ordered pairs of elements of V. So, if cardinality of V equal to n, cardinality of V x V is n square, every pair is involved in V x V.

$$|V| = n$$
$$|V \times V| = n^2$$

For example if v equal to {1 2 3}, V x V is 1 1, 1 2, 1 3, 2 1, 2 2, 2 3, 3 1, 3 2, 3 3; and this completes the description.

$$V = \{1,2,3\}, \quad V \times V = \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\}$$

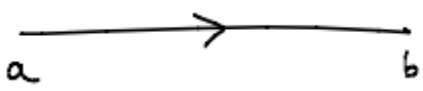
Now you can imagine V x V for any finite set, all pairs and each pair is an ordered pair, 1 2 is not equal to 2 1, this is a different order and this is a different order, both of them may involve 1 and 2, that is the difference between an ordered pair and unordered pair. Unordered pairs are like sets, it is not ordered therefore you have, set 1 2 is same as set 2 1.

$$(1,2) \neq (2,1)$$
$$\{1,2\} = \{2,1\}$$

Because in a set multiple copies do not matter, the order in which you write down the elements do not matter, because set is like a bag of items it is all there, that is all. But no specific order and anything just a collection of objects, okay. So set theoretically 1 2, 2 1 does not make any difference but the ordered pair, 1 2 is different from 2 1 okay, so E is a subset of the ordered pair. The vertex V is called vertex set and E is called edge set and an element a b belonging to E is called a directed edge, it is called directed edge.

$$(a, b) \in E$$

We write a directed edge like this.



This is a pictorial representation of a directed edge, this is the pictorial representation of a directed edge. The ordered pair is a,b, so we say that this edge, the edge a,b is said to leave a, they said it leaves a and it arrives b, said to leave a and said to arrive a, edge a,b

is an edge leaving a and arriving b. We indicate this in the picture by an arrowhead, indicates the order or direction. So every element of the edge set E is a directed edge and we represent pictorially by writing a line or sometime a simple curve, put a arrowhead indicating the direction, from where the edge leaves and to which vertex it arrives okay.

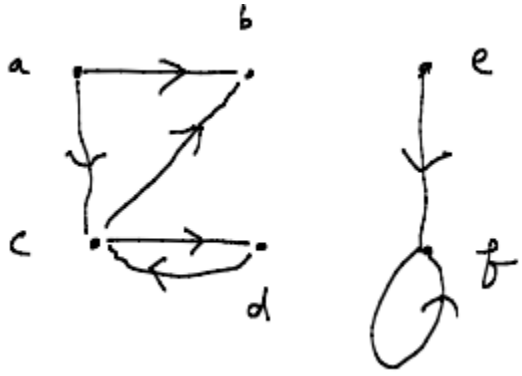
The vertices themselves, let us take an example, let V equal to {a, b, c, d, e, f} and E equal is the following ordered pair { (a,b), (a,c), (c,b) (c,d), (d,c), (f,f), (e,f) } V and a subset of V x V, so directed graph.

$$V = \{a, b, c, d, e, f\}$$

$$E = \{(a,b), (a,c), (c,b), (c,d), (d,c), (f,f), (e,f)\}$$

Although we can process it and handle it purely as sets, it is much more convenient and it is very easy to understand and build strong intuition if you work with a pictorial representation. In a pictorial representation, we mark each vertex with a point on the plane. So a is one point, b is another, c is another, d, e, f. So this is a, b, c, d, e, f.

You can mark the points anywhere. There is no restriction, just mark the points and label with the vertex elements. There are 6 vertices and there are 6 points. Now join by a line or a simple curve a b you join this and put an arrow here, a c join like this and put the arrowhead this way, from a it leaves a and arrives at c. c b - draw an arrow like this; c d, d c you can see that it is going the opposite direction also, c d is an edge, d c is another edge. f f, f to f this called the self loop and e f. So this is a pictorial representation of V.

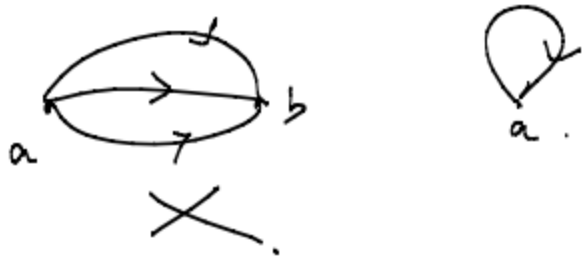


In the picture you could very clearly see and then understand various properties. When you look at the sets it is difficult to get an idea about the nature of the graph. The moment

you draw those pictures you can see that the graph has got 2 components or it is in 2 different pieces okay and from a you can go to b you can go to c but from c you cannot go to a there is no way you can go alright.

And if you want to go from a to d you can go from a to c and then from c to d you can travel you can traverse along the direction and then you can reach the vertices okay. Many things you understand about the graph when you see the pictorial version. So this is a pictorial representation. But this is useful only for humans to understand the nature and to explain and specify and then build some text or write something about it about the graph and so on. This is very helpful to build intuition and understanding, every vertex has got several edges leaving them for example if you take a from a there is one edge leaving for b another edge leaving for c okay. Therefore from a there are 2 edges that are leaving out that number is called the out degree of a vertex v is number of edges leaving v. How many edges are leaving from v? Similarly you have in degree number of edges arriving at v. For example a has got out degree 2. Out degree of a is 2 there are 2 edges that are leaving out but no edges coming into a therefore in degree of a is 0 okay.

Similarly in degree of e is 0 there is nothing that is coming in right. Suppose a directed graph is said to be simple if it has no self loops or multiple edges. between a pair of nodes. That means from a to b you do not have multiple edges you may have one and that is all.



The one going in the opposite direction is a different edge, okay so, and self loop.

We mostly deal with simple directed graph, but in case we have special requirements to deal with graph that I have got self loop we will say that explicitly. We will say that let G be a graph in which self loops are allowed, and so on, but for all our current purposes our graphs are going to be simple directed graph okay.

For a simple directed graph, we have the following interesting result Sigma v belongs to V of out degree of v is cardinality of E which we know is denoted by m, m is the number of edges.

$$\sum_{v \in V} \text{out\_degree}(v) = |E| = m$$

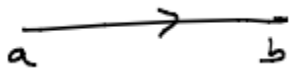
If you add all the out degrees, you get the count of number of edges. That is because if you take any vertex consider the edges that are leaving, every edge must leave some vertex, and it leaves from a vertex only once, therefore sum of all out degree.

Similarly, sigma v belong to V of in degree of v is cardinality of E that is equal to m, okay, in degree sum and out degree sum.

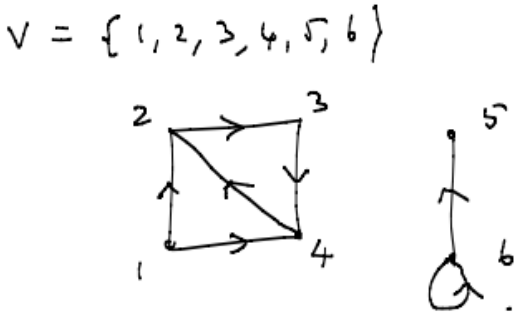
$$\sum_{v \in V} \text{in\_degree}(v) = |E| = m$$

Now we turn our attention to how to represent a directed graph in computers okay. We will give two options, both of them are popular and used in the programming in the coding process okay. In fact, we might even specify the algorithm based on the representation that we have chosen okay. We have an array based representation and that is called adjacency matrix representation.

If we have an edge a b, we say that b is adjacent from a, a is adjacent to b, in general a and b are adjacent, one next to another, that is adjacency alright.

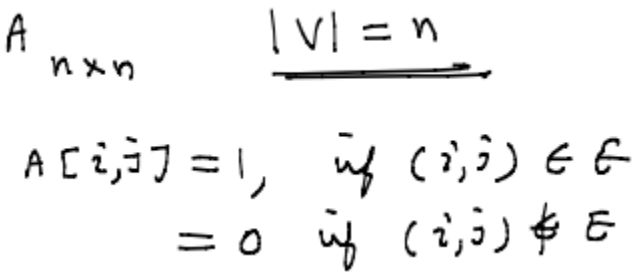


So we represent this information by a Boolean matrix, let us assume that V equal to 1, 2, 3.



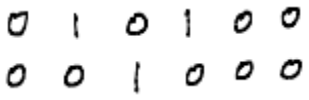
How will you represent using an adjacency matrix? We define the adjacency matrix, A of dimension n cross n, A is of dimension n cross n where cardinality of V is n. So each

matrix, the adjacency matrix will have n rows and n columns, where n is the size of the vertex set and A ij is equal to 1 if the edge ij belongs to E; equal to 0 if ij does not belong to E.

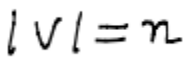


So, I can write this graph, we can represent this graph as a 6 by 6 matrix, okay, and in that 6 by 6 matrix I will have A12 as 1, 13 is 0, 14 as 1, 15 is not there, 16 is not there, 11 is not there, this is the first row.

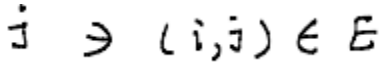
Second row, second row is with respect to 2, 2, 3 is there so 2, 1 is not there, 2, 2 is not there, 2, 3 is there, 2, 4 is not there, 2, 5 is not there, 2, 6. So in this way I can build a 6 by 6 matrix which has got zeros and ones.



A Boolean matrix one means, that particular pair is in the edge set, 0 means that pair is not in the edge set. So this is called the adjacency matrix representation. We will give adjacency list. In adjacency list representation, we are going to have an array of linked list, one for each vertex. We have an array of linked list. Therefore how many linked list we will have, we will have n linked list, that means we will have n linked list. Cardinality of V is n



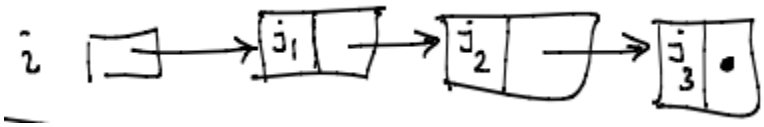
And in the linked list corresponding to vertex i all j values such that ij belongs to E will be listed out.



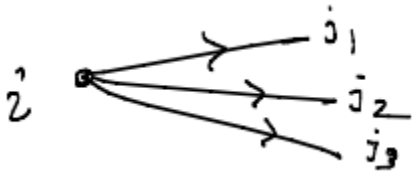
That is why it is called the adjacency list. The list corresponding to i contains all j such that ij belongs to E.

$$(i, j_1), (i, j_2), (i, j_3) \in E$$

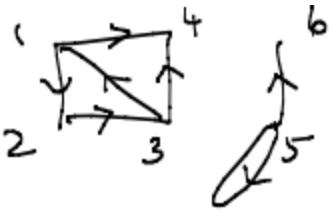
So for example ij1, ij2, ij3 if they belong to E then in the linked list corresponding to i, i will have i1, connected with i2 connected with i3 and then this is the last one, this is a null pointer okay, the last box has no further connections, you put a dot there to indicate that the linked list ends there.

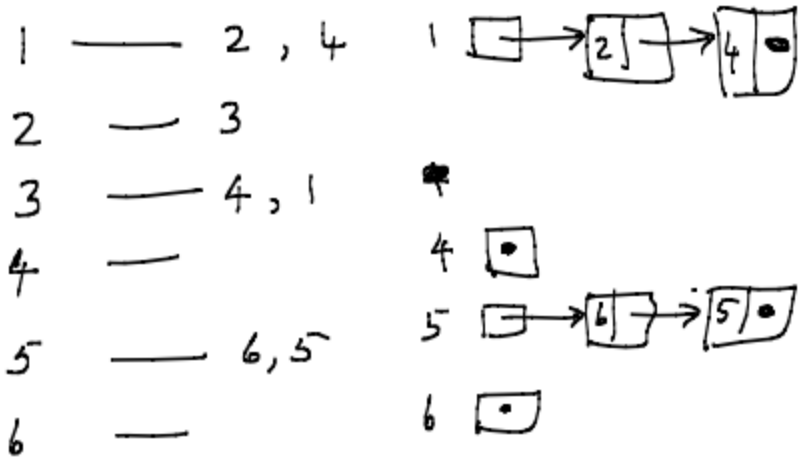


So all neighbors, all vertices to which you have arrived at from i, from i you have arrived at j1, from i you have arrived at j2 by an edge, so pictorially here you have i j1 j2, oh sorry not i j1 j2 j3, I made a mistake here let me correct it not i1. i has j1 j2 j3, i is pointing to first box that has j1, second box j2, third box j3, ij1, ij2, ij3, this is the pictorial representation, this is the linked list representation.



So let me take us into 1, 2, 3, 4, 5. Adjacency list for 1 consists of 2 and 4, you can write the boxes and for example from 1 you go to 2, you go to 4 you end there but I write like this from 2 you can reach 3 from 3 you can go to 4 and you can go to 1 from 3 there is an edge going to 1 there is an edge going to 4, from 4 nothing is going out okay, everything is coming in but from 4 nothing is going out therefore this is an empty list, so this is for 1. This represents an empty or you can use the same notation. The linked list is not referring to any box. It is a null pointer. It just ends there. It is not referring to any, the pointer is null pointer. 4 refers to a box it does not refer to any box it is referring to null that is it okay. 5 is pointing to 6 and also to 5 again 6 has got a null pointer, 5 has it is pointing to 6 which is pointing to 5 and that is the end of the list so in this way you can write the linked list representation.





This is directed graph what we have seen so far is a very basic definition of directed, we can have similar definitions for undirected graph however we will not have out degree and in degree we will have simply the notion of degree okay undirected graph. Alright we stop our discussion at this point. We will take a look at the basics of undirected graph in our next session. Thank you.