



# IIT Madras

ONLINE DEGREE

**Programming, Data Structures and Algorithms using Python**  
**Professor Madhavan Mukund**  
**Chennai Mathematical Institute**  
**Introduction to graphs**

So, having seen functions on arrays and lists, let us move to graphs, which are very interesting objects to study in computing.

(Refer Slide Time: 00:18)

The slide is titled "Visualizing relations as graphs" and features the IIT Madras logo. It contains a list of definitions and a diagram illustrating a relation between teachers and courses.

**Teachers and courses**

- $T$ , set of teachers in a college
- $C$ , set of courses being offered
- $A \subseteq T \times C$  describes the allocation of teachers to courses
- $A = \{(t, c) \mid (t, c) \in T \times C, t \text{ teaches } c\}$

**Teachers and courses**

The diagram shows a bipartite graph with two sets of nodes. On the left, the teachers are Sheila, Aziz, Priya, Kumar, and Deb. On the right, the courses are Biology, English, History, and Maths. Arrows represent the relation  $A$ : Sheila is connected to Biology; Aziz is connected to English; Priya is connected to English; Kumar is connected to History and Maths; and Deb is connected to Maths. The nodes for Sheila and Kumar are circled in red.

So, we know that a graph is a useful way to visualize a relation. So, let us imagine this relation. So, we are talking about a class in which you have or an institution which you have teachers and courses. And you have some allocation of courses to teachers. So, you have two types of objects, you have teachers who are people and you have courses which are to be taught. And you can describe this allocation as a relation.

So, you can say,  $A$  the allocation is a subset of  $T$  cross  $C$ , it is a binary relation, which consists of pairs or the form  $T$  comma  $C$ , where  $T$  is a teacher, and  $C$  is a course. So, there are many possible pairs  $T$  cross  $C$ . And we are not taking all of them, but some of them, so it is a binary relation. And what we see on the right is a representation of this as a graph. So, right now, it is very clear that we have these nodes.

So, we have one node for every object of interest. In this case, we have two different types of objects. But in the graph, we do not mean we label them maybe, but we represent them in the same way as black dots. So, on the left-hand side, we have the teachers on the right-hand side, we have the courses. And then we draw these edges, these arrows from a teacher to a

course. So, there is a direction to this edge and this arrow indicates that this teacher is going to be teaching that course.

And of course, there could be situations where two different teachers teach the same course. So, this could happen, you could also have a situation where the same teacher is teaching two different courses, which is not captured in this particular example. But of course, you could have that somebody is teaching both biology and history, for instance.

(Refer Slide Time: 01:54)

The slide is titled "Visualizing relations as graphs" and features two graphs illustrating different types of relationships.

**Teachers and courses:** This graph shows a set of teachers  $T$  and a set of courses  $C$ . The relationship is represented by a directed edge from a teacher to a course, indicating that the teacher teaches that course. The set of edges is defined as  $A = \{(t, c) \mid (t, c) \in T \times C, t \text{ teaches } c\}$ .

**Friendship:** This graph shows a set of students  $P$ . The relationship is represented by an undirected edge between two students, indicating that they are friends. The set of edges is defined as  $F = \{(p, q) \mid p, q \in P, p \neq q, p \text{ is a friend of } q\}$ . The slide also notes that  $(p, q) \in F$  if and only if  $(q, p) \in F$ , indicating that friendship is a symmetric relation.

The slide includes a video feed of the presenter, Madhavan Mukund, and a navigation bar at the bottom with the text "Madhavan Mukund", "Graphs", and "DOCA".

Here is another typical relationship that we describe using graph. So, supposing you have a group of people, and some of them are friends with others, then the notion of being a friend, again, describes certain pairs, so some pairs of people are friends, and some pairs are not friends. So, those who are friends, you connect by an edge. And now, we can assume that friendship is a symmetric relation. So, if I am somebody is friend, then that person is also my friend.

So, therefore, if  $p$  is a friend of  $q$ , then  $q$  should be a friend of  $p$ . So, then we have a graph like this, where now technically speaking, we have arrows representing pairs in both directions. So, we have an arrow from  $P$  to  $Q$ , saying that  $Q$  is a friend of  $P$ , and we have an arrow from  $Q$  to  $P$ , saying that  $P$  is a friend of  $Q$ . And because we have arrows always in both directions, we just ignore them and draw only a single edge.

(Refer Slide Time: 02:46)

## Graphs



- Graph:  $G = (V, E)$ 
  - $V$  is a set of vertices or nodes
    - One vertex, many vertices
  - $E$  is a set of edges
  - $E \subseteq V \times V$  — binary relation **IRREFLEXIVE**

Navigation icons

Madhavan Mukund


Graphs

02:46

So, formally, a graph is a set of vertices, those are the black dots. So, they are called vertices, or sometimes they are called nodes. And we take them and connect them using edges. So, this edge technically is a binary relation, it says some pairs of vertices, so there is a whole, all the pairs of vertices are  $V \times V$ , some pairs of vertices are connected by edges. And usually, we have the constraint that a vertex is not connected to itself. so that is usually one of the constraints that we have. So, we usually assume that this is in the relational terminology **IRREFLEXIVE**.

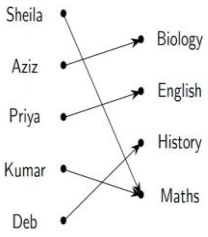
(Refer Slide Time: 03:23)


### Graphs



- Graph:  $G = (V, E)$ 
  - $V$  is a set of vertices or nodes
    - One vertex, many vertices
  - $E$  is a set of edges
  - $E \subseteq V \times V$  — binary relation
- Directed graph
  - $(v, v') \in E$  does not imply  $(v', v) \in E$
  - The teacher-course graph is directed

Teachers and courses






Madhavan Mukund

Graphs

So, this graph can have typically will be directed. So, I will have edges from, say, P to Q, and I need not have for every such edge, an edge back from Q to P. So, a very particular example of a directed graph was the thing with teachers and courses, because the edges all point from teachers to courses, it does not make sense in such an interpretation to think of an edge starting at a course because a course cannot teach something else. So, these edges represent who teaches what, so the left-hand, left-hand side has to be a who the right-hand side has to be a what.

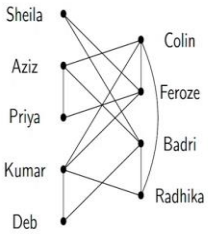
(Refer Slide Time: 03:57)


### Graphs



- Graph:  $G = (V, E)$ 
  - $V$  is a set of vertices or nodes
    - One vertex, many vertices
  - $E$  is a set of edges
  - $E \subseteq V \times V$  — binary relation
- Directed graph
  - $(v, v') \in E$  does not imply  $(v', v) \in E$
  - The teacher-course graph is directed
- Undirected graph
  - $(v, v') \in E$  iff  $(v', v) \in E$
  - Effectively  $(v, v')$ ,  $(v', v)$  are the same edge
  - Friendship graph is undirected

Friendship





Madhavan Mukund

Graphs

On the other hand, the friendship graph that we constructed, as we said, is symmetric. So, we have edges between two pairs in both directions or not at all, we never have a situation where

P is a friend of Q, but Q is not a friend of P. And in such situations, we will just ignore the edges, and just draw these with these lines connecting the vertices. So, basically an undirected graph is a graph in which all the edges are symmetric.

So, we can effectively think so we will think still represented by edge in both directions because we can traverse this edge in both directions. So, if you want, you can think of these as roads. So, if these are two locations served by a road, if you have a directed road, then you can only go from one direction from one to the other, you cannot come back on that road. Whereas if you have a bi-directional road, then it is an undirected road, you can go in either direction.

(Refer Slide Time: 04:47)

**Paths**

- A **path** is a sequence of vertices  $v_1, v_2, \dots, v_k$  connected by edges
  - For  $1 \leq i < k, (v_i, v_{i+1}) \in E$
- Normally, a path does not visit a vertex twice
- A sequence that re-visits a vertex is usually called a **walk**
  - Kumar — Feroze — Colin — Aziz — Priya — Feroze — Sheila

**Friendship graph**

Sheila, Colin, Feroze, Badri, Radhika, Priya, Kumar, Deb, Aziz

Madhavan Mukund | Graphs

So, one of the key things that we are interested in graphs are sequences of edges, which take us from one node to another node and this is what is called a path. So, here we see two paths, we see this red path which takes us from the node representing Priya to the node representing Radhika. And the other one is a blue path which also connects Priya to Radhika but through a different sequence of intermediate nodes.

So, these could represent, for instance, routes by which some information that Priya wants to convey or in some object, supposing Priya wants to send something which is precious not to be sent by courier or post by hand to Radhika, then Priya has to find some of her friends who can, in turn, find their friends and so on, who finally will be able to find someone who knows Radhika can pass it on. And here we see that there are at least two different ways to do this.



So, normally, when we are talking about paths in a graph, a path does not visit a vertex twice. So, here, for instance, we have this red path here or red sequence of edges, which certainly could be seen as a sequence where every consecutive thing is part of the edge relation. So, we start for instance here.

So, we start here, and we go from Kumar to Feroze. And then Feroze to Colin, then Colin to Aziz, then Aziz to Priya, then we come back to Feroze. And then we come here. So, this is technically a sequence of edges, each of which extends the previous sequence by a valid point, but we are not going to call this apart, when we encounter such things, we will usually call it a walk.

(Refer Slide Time: 06:24)

**Reachability**

- Paths in directed graphs
- How can I fly from Madurai to Delhi?
  - Find a path from  $v_9$  to  $v_0$
- Vertex  $v$  is **reachable** from vertex  $u$  if there is a path from  $u$  to  $v$

**Airline routes**

Graph showing vertices  $v_0$  through  $v_9$  and directed edges. A red path is highlighted from  $v_9$  to  $v_0$ .

Madhavan Mukund      Graphs

So, paths in directed graphs, obviously have to respect the arrows. So, if I have, for instance, an airline route, so this is a typical thing that when seen in real life and graphs, you see these route maps of trains or planes, and they describe which pairs of cities are connected by flights. So, let us assume that these are some flights for an airline operating in India and say that  $V_0$  represents Delhi and  $V_9$  represents Madurai, then what we want to know is whether we can take flights within this airline and travel from Madurai to Delhi, which means, Can I take a sequence of edges starting at  $V_9$ , and reach  $V_0$ .

And here, for instance, there is this red sequence of edges that you can see the ones that have been drawn here, so this red sequence of edges does represent one way of taking flights, a long sequence of flights, in this case, 1, 2, 3, 4, 5 flights to get from Madurai to Delhi.

(Refer Slide Time: 07:21)

**Reachability**

■ Paths in directed graphs

■ How can I fly from Madurai to Delhi?

- Find a path from  $v_9$  to  $v_0$

■ Vertex  $v$  is **reachable** from vertex  $u$  if there is a path from  $u$  to  $v$

■ Typical questions

- Is  $v$  reachable from  $u$ ?
- What is the shortest path from  $u$  to  $v$ ?
- What are the vertices reachable from  $u$ ?
- Is the graph **connected**? Are all vertices reachable from each other?

**Airline routes**

Madhavan Mukund

Graphs

**Reachability**

■ Paths in directed graphs

■ How can I fly from Madurai to Delhi?

- Find a path from  $v_9$  to  $v_0$

■ Vertex  $v$  is **reachable** from vertex  $u$  if there is a path from  $u$  to  $v$

■ Typical questions

- Is  $v$  reachable from  $u$ ?
- What is the shortest path from  $u$  to  $v$ ?
- What are the vertices reachable from  $u$ ?
- Is the graph **connected**? Are all vertices reachable from each other?

**Airline routes**

Madhavan Mukund

Graphs

So, if this happens, then we say that the target is reachable, so it could happen that things are reachable, it could happen that things are not reachable. So, if they are reachable, then one of the questions that we might ask is among all the different ways of reaching what is the shortest way. Other thing you might ask is, where all can I reach? So, there are many different questions depending on the context that you might ask about reachability.

And of course, one of the things that you might want to know is this graph is connected? Are all the vertices reachable from each other? Or do you get stuck somewhere? So, if we look at this graph, for instance, if you take this edge from  $V_4$  to  $V_3$ , and you put it in only one direction, then I claim that you can still reach from everywhere to everywhere because earlier, you could go directly from  $V_3$  to  $V_4$ .



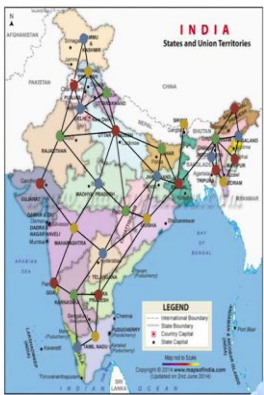
But now, you can still do the same thing by going around. So, if you want the what is now missing is the ability to go from V3 to V4, because I removed this edge red, so this edge is gone. But now, I can still go from V3 to V6, V6 to V5, V5 to V7, and V7 to V4. So, the ability to go from V3 to V4 has not changed, it has become a little more tedious, but it is not impossible. So, this graph, if it was connected earlier, remains connected, even though we have removed one edge.


On the other hand, supposing we take this edge between V4 and V0 and we make it in a single direction. So, earlier, I could come from V0 to V4. Now, if I remove this, the question is, can I come? Well, if I start at V4, I can go to V1, from V1 I can go to V2, from v2 I can go back to V0, but I can go nowhere else. So, if I am now in this upper triangle of V0, V1, and V2, I stuck there, I cannot come to the lower part of the graph. So, now the graph is not connected anymore. So, the question that we want to ask is, is the graph that is presented to us connected or not? And this can have many implications.

(Refer Slide Time: 09:15)

### Map colouring

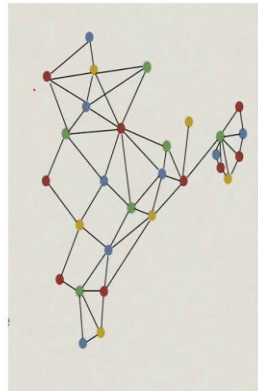
- Assign each state a colour
- States that share a border should be coloured differently
- How many colours do we need?
- Create a graph
  - Each state is a vertex
  - Connect states that share a border
- Assign colours to nodes so that endpoints of an edge have different colours





Madhavan Mukund
Graphs

- Assign each state a colour
- States that share a border should be coloured differently
- How many colours do we need?
- Create a graph
  - Each state is a vertex
  - Connect states that share a border
- Assign colours to nodes so that endpoints of an edge have different colours
- Only need the underlying graph
- Abstraction: if we distort the graph, problem is unchanged



Madhavan Mukund

Graphs

So, reachability is not the only kind of question that we are interested in graphs there are many other kinds of things which can be modelled and solved as graphs. So, for instance, here is a problem which superficially does not seem to require graphs, which is to colour a map properly. So, when we colour a map properly, we want to neighbouring states or to neighbouring countries, which share a border to have different colours.

So, one of the questions that you can ask is how many colours do we need. So, to transfer this to a graph problem, what we do is we first create a vertex for every state that needs to be coloured. And now we want to express this constraint that neighbouring states should have different colours. So, we have to capture this neighbouring relation. So, we do that by connecting them together. So, we attach an edge between every pair of states, which shares a border.


So, now we have a graph. And now what is our constraint, our constraint is we must assign a colour to every one of these black dots such that if two black dots are connected by an edge, they must have different colours. So, we must assign colours to nodes so that the endpoints of an edge have different colours, you cannot have a green on one side and the green on the other side, because that would mean two neighbouring states have the same colour. So, this is called the map colouring problem.

But notice that by coming up with this graph representation, we have abstracted the problem so that we are now no longer worried about the shapes of the states, the sizes of the states, the kind of orientation of the boundary, how much of a boundary do they share? All we are interested in is this picture, which is, what are the underlying states and how are they connected?

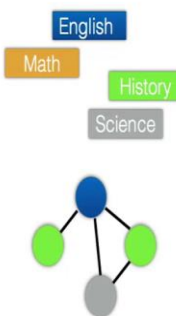
And once we have this, it does not even need to respect that true physical geography. So, I can take some of these nodes and move them aside, I can make the graph look different superficially, without changing the underlying nature that is relevant to the problem namely, which points are neighbours of which other points.

(Refer Slide Time: 11:17)

Graph colouring



- Graph  $G = (V, E)$ , set of colours  $C$
- Colouring is a function  $c : V \rightarrow C$  such that  $(u, v) \in E \Rightarrow c(u) \neq c(v)$
- Given  $G = (V, E)$ , what is the smallest set of colours need to colour  $G$ 
  - **Four Colour Theorem** For planar graphs derived from geographical maps, 4 colours suffice
  - Not all graphs are planar. General case? Why do we care?
- How many classrooms do we need?
  - Courses and timetable slots, edges represent overlapping slots
  - Colours are classrooms




Madhavan Mukund

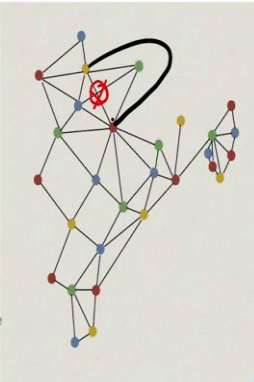
Graphs

---

Map colouring



- Assign each state a colour
- States that share a border should be coloured differently
- How many colours do we need?
- Create a graph
  - Each state is a vertex
  - Connect states that share a border
- Assign colours to nodes so that endpoints of an edge have different colours
- Only need the underlying graph
- Abstraction: if we distort the graph, problem is unchanged



Madhavan Mukund

Graphs

So, this is called the graph colouring problem. So, we are given a graph and we are given a set of colours. And what we want to do is assign a colour to every vertex such that no pair of adjacent vertices, there is no vertices which are connected by an edge have the same colour. So, this is what is saying, if  $u, v$  are edges in a, if  $u, v$  is an edge, then the colour assigned to  $u$  should not be the colour assigned to  $v$ .

So, the question that we were asking in that map colouring problem is, what is the smallest set of colours that we need? So, there is a very famous theorem in graph theory, which says that if you have what is called a planar graph, so planar graph is something which you can draw without any crossing edges. So, if you go back to the graph that we had before, you will notice that in this thing, there are no, so here we have an edge, which is crossing.

But actually, you can take this edge, and you can drop it, and you can instead draw the same edge. So, remember, we said that we can modify this graph, so long as we do not change anything, which is intrinsic to the graph. So, we can take that edge and move it to the outside of the graph. Edges do not have to be straight lines, they are just something which connects vertices together.

And now, I claim that we have the same graph in which no two edges cross on the paper. So, if you can draw a graph like this, this is what is called a planar graph. And it is not difficult to see that if you draw a map a physical map of countries or states, and you take the graph that we constructed based on the borders, it is going to be a planar graph. And it turns out that for planar graphs, graphs which can be drawn without having edges crossing each other, you can colour such graphs, always with four colours. If it is not planar, this is not true.

And then it becomes a computationally interesting question to ask, what is the minimum number of colours? And it is one of these problems for which there is no efficient algorithm known? So, you might ask, where do we need to apply this? Because colouring seems to be derived from this map question. And for the map question, we know the answer is 4 because it is planar.

So, why do we need to worry about non-planar graphs? So, the thing is, there are many other questions which can be reduced to graph colouring. So, here is one. So, supposing you are running a school or a college, and you need to allocate classrooms, for the classes that are running. So, you have a timetable. So, in this timetable, this is kind of the time is shown from left to right.

So, at the beginning of the day, you have a math class before the math class ends and English class starts before the English class ends, History and Science start, but notice that Science starts after Maths ends. So, the science class does not overlap with a math class. So, now we have four different classes which run during the day. But History and Science both run after Maths, so they both do not overlap with Maths.

The question that we want to answer now in this particular scenario is how many classrooms will we need so that no two classes are scheduled in the same room at the same time. So, obviously, if we allocate one classroom per class, there is no problem. So, we could have four classrooms for these four classes. So, we could have an orange classroom for the Maths class and a green classroom for the History class and so on.

So, clearly, if we have as many classrooms as there are classes, there is no question but in almost any institution, the number of classes or courses which are taught will obviously exceed the number of rooms that are available. So, one of the questions that is interesting is to ask, can we manage with the rooms that we have, so what is the minimum number of rooms that we need in order to schedule the classes the way we have done and not have this overlap.

So, in this case, we can think of this edge relation. So, the classes that have to be taught at the vertices, the edge relation represents two classes which overlap, and therefore, they must fall into different classrooms, because they cannot share a classroom. The classrooms themselves are the colours. So, now if two edges or two classes are connected by an edge, then they cannot be assigned the same classroom.

So, if classrooms are colours, they cannot be assigned the same colour. So, the edges here represent this overlap, so that we mentioned above, so these edges, so for notice that there is no overlap between these two pairs. So, there is no edge here. And there is no edge here because Maths ends before History and Science both start, but History and Science overlap with each other.

Now, as for our colouring algorithm, we are allowed to use the same colour for this node as either this node or this node because they are not connected. So, one possible way of reducing the number of classrooms here is to reuse the green classroom for Maths that we are going to later use History. We could also equivalently reuse the grey classroom that we use for Science. So, this is a situation where we can make do with three colours, even though we have four classes.


And of course, if you have a more complicated trade you, then it is a more interesting question to ask how many minimum classrooms you need to have in order to fit that schedule, or you have to change the schedule to adapt to the number of classes because of course, you cannot change the number of rooms very easily. So, you can change the schedule.




So, you might have to change the schedule in order to make it fit within the infrastructure that you have.

(Refer Slide Time: 16:36)

Vertex cover



- A hotel wants to install security cameras
  - All corridors are straight lines
  - Camera can monitor all corridors that meet at an intersection
- Minimum number of cameras needed?



Madhavan Mukund

Graphs


So, let us look at another problem. So, here is a problem, which is to do with security cameras. So, as you know, security cameras are placed in such a way that they capture as much of the surroundings as they can and you want to minimize the number of security cameras. So, supposing you are in a hotel and the hotel has corridors, you would ideally place your cameras at intersections of corridors, let us assume that the camera can rotate around and see all the corridors which meet at a point.

So, a camera can monitor all the corridors. So, if supposing you have a lift here, and then after you come out of the lift you have say corridors which look like this, then if you place a camera at this point it can see all these three corridors. So, this is the kind of placement that we want. So, what is the minimum number of cameras that you need, if you know the layout of the corridors in your hotel.

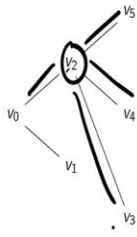



(Refer Slide Time: 17:26)

Vertex cover



- A hotel wants to install security cameras
  - All corridors are straight lines
  - Camera can monitor all corridors that meet at an intersection
- Minimum number of cameras needed?
- Represent the floor plan as a graph
  - $V$  — intersections of corridors
  - $E$  — corridor segments connecting intersections





Madhavan Mukund


Graphs

17:26

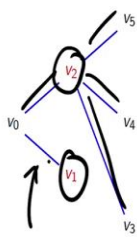
So, here we would like to represent the floor plan as a graph. So, the intersections now become the meeting points of these corridors. So, the actual intersections of your corridors become the vertices. And now from each vertex, we said that if we put a camera here, for instance, it can monitor all these three, all these four corridors. So, the edge represents a corridor, which is incident at an intersection, and this edge says that there is a corridor, which goes from intersection  $V_2$  to intersection  $V_3$ . And now the question is, where should we put our cameras?


(Refer Slide Time: 18:07)

Vertex cover



- A hotel wants to install security cameras
  - All corridors are straight lines
  - Camera can monitor all corridors that meet at an intersection
- Minimum number of cameras needed?
- Represent the floor plan as a graph
  - $V$  — intersections of corridors
  - $E$  — corridor segments connecting intersections
- Vertex cover
  - Marking  $v$  covers all edges from  $v$
  - Mark smallest subset of  $V$  to cover all edges





Madhavan Mukund

Graphs

18:07

So, this says, we must choose some vertices. So, supposing I choose  $V_2$ , then  $V_2$  covers all of these. But I want to monitor every corridor in my hotel. So, from  $V_2$ , I cannot monitor this

particular corridor. So, I need to add one more camera. So, I could put like  $V_0$  or at  $V_1$ . So, let me put it  $V_1$ . So, if I now place my cameras at  $V_2$  and  $V_1$ , then every corridor on this map is monitored. So, this is what is called a vertex cover, it is a set of vertices, such that every edge has one endpoint in the set.

So, either one end or the other end of the edge must be in the set. So, every endpoint every corridor is monitored by one of these cameras. And the question here computationally is again, of course, I can put a camera everywhere. And then I will obviously cover all the corridors. But what is the smallest number of intersections that I need to place cameras at or in other words, what is the smallest vertex cover that I can achieve?

(Refer Slide Time: 19:03)

**Independent set**

- A dance school puts up group dances
  - Each dance has a set of dancers
  - Sets of dancers may overlap across dances
- Organizing a cultural programme
  - Each dancer performs at most once
  - Maximum number of dances possible?
- Represent the dances as a graph
  - $V$  — dances
  - $E$  — sets of dancers overlap

Madhavan Mukund      Graphs

A related problem is what is called an independent set. So, here, let us look at this scenario. So, we have a kind of Fine Arts Academy, which can put up various cultural programs including some group dances. Now, in a group dance, of course, a number of people are involved. And it could be that some dancers are trained to dance and more than one group dance. So, suppose that this academy wants to put up a cultural program, each dance requires the person performing to wear some elaborate costume.

So, it is not feasible for a dancer to take part in two dances in the same evening because he or she will have to change their costume and that will be very problematic. So, now, you want to know, given the performers who are available and the dances that are available, what is the maximum number of dances you can schedule in this program, so that you never have to take a dancer from one dance, and have them change their costume and take part in another dance.

So, in this case, again, you can represent each of the dances as a vertex. And you can say that two dancers are connected if they need one person to participate in both, so if they have a common member in that group, then they are connected. So, here it says that  $V_3$  and  $V_7$  for instance, somebody will be required, who will participate in both of them. Similarly, say  $V_5$  and  $V_6$ , but there is no connection between  $V_6$  and  $V_8$ . So, anybody who takes part in  $V_6$  is guaranteed not to be part of  $V_8$  and so on. So, in this case, we want to find out what is the maximum number of dances we can simultaneously schedule on a day so that there is no such overlap.

(Refer Slide Time: 20:44)

### Independent set

- A dance school puts up group dances
  - Each dance has a set of dancers
  - Sets of dancers may overlap across dances
- Organizing a cultural programme
  - Each dancer performs at most once
  - Maximum number of dances possible?
- Represent the dances as a graph
  - $V$  — dances
  - $E$  — sets of dancers overlap
- Independent set
  - Subset of vertices such that no two are connected by an edge

Madhavan Mukund
Graphs

So, here is one such allocation, supposing we choose  $V_4$ , then we cannot schedule these dances because they have some overlap. On the other hand, there is no overlap with  $V_7$ , so we can take  $V_7$ , so  $V_7$  will rule out  $V_6$ , but it does not rule out  $V_5$ . And if we have done  $V_4$ ,  $V_5$ , and  $V_7$ , we still have the possibility of  $V_2$  because  $V_2$  is not connected to any of these.

So, these vertices, now unlike our previous situation, where the vertices are trying to cover all the edges, here, we are saying these vertices should be mutually disjoint from each other, no two vertices that we choose, no two dances that we choose must be connected because they should not share a participant.

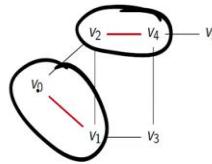
So, in this case, we have found four such and this is called an independent set. So, we want the largest number of dances earlier, we wanted the smallest number of cameras. So, that was the smallest vertex cover. Here, we want the largest number of dances that we can choose simultaneously in a day is program. So, this is the maximum independent set.

(Refer Slide Time: 21:44)

## Matching



- Class project can be done by one or two people
  - If two people, they must be friends
- Assume we have a graph describing friendships
- Find a good allocation of groups
- **Matching**
  - $G = (V, E)$ , an undirected graph
  - A matching is a subset  $M \subseteq E$  of mutually disjoint edges
- Find a maximal matching in  $G$



Finally, let us look at a very important problem, which is called Matching. And suppose we have a class project we are going to assign. And we allow groups of two students to do this project together. But we know that for two students to do this together, they must be friends, if they do not get along with each other, there is no point assigning them to a group.


So, we have a graph like the one on the right describing the friends. So, it says that  $V_2$  is a friend of  $V_4$ , but for instance,  $V_2$  is not a friend of  $V_3$ , because there is no edge there. What we want to know is how many groups can be formed from this, so that we only pick pairs, which are friends. So, we want to find a good allocation. So, this is a subset. So, the allocation that we pick will be a subset of the edges.

And once we pick this subset, those will form the groups and if we have any person who is not paired up with another person, that person will have to work on their own. So, maybe this is a complicated project. And we would ideally like most people to work in groups, we want to find the largest such. So here, for instance, is what is called a maximal matching. So, I have paired up these two, I have paired up these two, I cannot pair up now 5 with anybody because the only thing five can be paired up with this 4 who is already part of another pair and same with 3, 3 can only be paired up with 1 and 4.


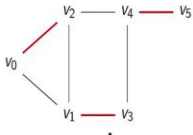
So, this matching is maximal in the sense that I cannot add any more edges and make it a larger matching from what I have right now. But it turns out that this is not a maximum matching the sense that this says we can form two groups of two and then two people are left alone, but maybe in the six people we can form three groups of two so that everybody has a pair and yes, indeed it is possible.

(Refer Slide Time: 23:26)

### Matching



- Class project can be done by one or two people
  - If two people, they must be friends
- Assume we have a graph describing friendships
- Find a good allocation of groups
- **Matching**
  - $G = (V, E)$ , an undirected graph
  - A matching is a subset  $M \subseteq E$  of mutually disjoint edges
- Find a maximal matching in  $G$
- Is there a **perfect matching**, covering all vertices?




Madhavan Mukund      Graphs


So, we can pair up 5 to 4, 3 to 1, and 0 to 2 as we see here. Now everybody is assigned a pair and now everybody is covered. So, in this particular case is quite easy to say that if you have an even number of vertices and you have half as many edges in the matching that everybody is covered. So, in general, this could be a question which is if I give you such a graph, and I asked you to match up pairs with this constraint, then what is the largest matching that you can construct.

(Refer Slide Time: 23:56)

### Summary



- A graph represents relationships between entities
  - Entities are vertices/nodes
  - Relationships are edges
- A graph may be directed or undirected
  - A is a parent of B — directed
  - A is a friend of B — undirected
- Paths are sequences of connected edges
- Reachability: is there a path from  $u$  to  $v$ ?
- Graphs are useful abstract representations for a wide range of problems
- Reachability and connectedness are not the only interesting problems we can solve on graphs
  - Graph colouring
  - Vertex cover
  - Independent set
  - Matching
  - ...



Madhavan Mukund      Graphs

So, to summarize, graphs are very interesting because they express relationships, many flexible types of relationships in a very concise way. Graphs can be directed as we saw or undirected. So, when an undirected graph really every edge has a matching edge in the



reverse direction, so we can think of it as a single edge which has two directions associated with it. One of the most fundamental questions that we need to ask about graphs is reachability.

Can I get from one node to another node by a path which is a sequence of connected edges? But there are a wide range of other problems also, which are not directly called connected to reachability which are also interesting. So, we saw graph colouring, we saw vertex cover, we saw the independent set problem, we saw matching. So, these are all a variety of problems, which are interesting computation problems on graphs and which have a number of practical implications. So, we will study some of these as we go along.

