

Cover of an Element

Let (S, \leq) be a poset and $x, y \in S$. Then y is a cover of x , if:

- ❖ $x < y$
- ❖ There is no $z \in S$, with $x < z$ and $z < y$

❖ 2 covers 1

❖ 6 does not cover 1

- ❑ An element may not have a cover
 - ❖ Ex: 8, 12
- ❑ An element may have more than one cover
 - ❖ Ex: 2, 3 covers 1
- ❑ An element may cover many elements
 - ❖ Ex: 6 covers both 2 as well as 3

So, imagine you are given an arbitrary poset less than equal to relationship. This is not again a numerical less than equal to, this is an arbitrary relation, R , which is reflexive, anti symmetric and transitive. Then if I take a pair of elements x, y then the element y is called the cover of element x if the following two conditions hold. The element x should be related to the element y and of course $x \neq y$, that is why the less than symbol. And there should not exist any intermediate element $\exists z, x \leq z$ and $z \leq y$.

So, pictorially, you can imagine that y is a cover of x if I view the Hasse Diagram then in the when I go from bottom to up y is immediately occurring or y is occurring on top of x layer wise and there is no intermediate element or no element z in the intermediate layer. So, for instance here in this Hasse diagram the element 2 covers the element 1 because in between 2 and 1 there is no intermediate element. You have the element 1 which is related to the element 2 and between 1 and 2 there is no intermediate elements. But the element 6 does not cover the element 1, even though the element 1 is less than 6, because element 1 is indeed the related to 6 as per this Hasse diagram. But in between 1 and 6 you have this element 3 such that 1 is related to 3 and 3 is related to 6. So, that is why 6 will not be considered as a cover of 1, but 3 can be considered as a cover of 1 because in between 3 and 1 there is no intermediate element.

So, it turns out that in a partially partial order set every element need not have a cover. So, for instance, if you take the Hasse diagram on your left-hand side the elements 8 the element 12, it

does not have any common. There is no element on top of 8, there is no element on top of 12. Similarly, an element, we have more than one cover. So, as I said earlier both 2 and 3 covers 1. And an element may cover multiple elements. So, for instance here, in this Hasse diagram or in this poset 6 covers 2 as well as 3. So, these are the some of the properties of the cover of an element. (Refer Slide Time: 26:01)

Maximal and Minimal Element

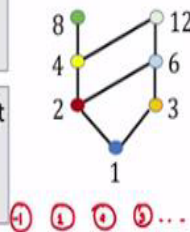
□ Let (S, \leq) be an arbitrary poset and $a \in S$

❖ a is called a **maximal element**, if it has no cover

- There is **no** $b \in S$, with $a < b$
- Ex: 8, 12 are the maximal elements

❖ a is called a **minimal element**, if it covers no element

- There is **no** $b \in S$, with $b < a$
- 1 is the minimal element



□ Every poset has **at least one** maximal and one minimal element

□ An element of a poset can be **both** maximal as well as a minimal element

❖ Ex: (\mathbb{Z}, \leq) , where \leq is the "equal-to" relationship

$(1,1) (2,2) (3,3)$
 $(-1,-1) (-2,-2)$

Let us next define what we call as the maximal and minimal element in a poset. So, if you are given an arbitrary poset and an element a from the set S . Then the element a is called as the maximal element if it is on the top most layer informally, or in a loose sense or if it has no cover. More formally, a is called maximal element, $\exists b \in S, b < a$ i.e., is no element b on top of a that means there is no element b such that a is related to b where a is different from b .

So, if I take this poset, 8 and 12 are both maximal elements. Because there is no element on top of a or no element b such that 8 is related to that b . There is no element b such that 12 is related to that element b or so. There is 4 will not be called a maximal element sorry and 6 cannot be called a maximal element. Why 4 cannot be called a maximal element. Because 4 is related to 8, there is something on top of 4 and so on.

Similarly, I can define what we call as a minimal element. So, an element is called as a minimal element if it occurs at the lower level of your Hasse diagram or in other words, it has no if it covers

no element. Namely, a is called minimal element, $\exists b \in S, a < b$ i.e., there is no element b in your set S which occurs below a or such that b is related to a . So, for instance the element 1 here is the minimal element. Because there is no element b for the down 1 in your Hasse Diagram such that b is related to 1.

That tells you that why when we constructed the Hasse diagram, we assume that arrows are pointed from bottom to up. That helps us to understand these notions of maximal element and minimal element in an easy fashion. Now, it is easy to prove that if you have a poset over a non-empty set. I forgot to mention here over a non-empty set, then it has at least one maximal element and one minimal element.

So, for instance if the poset is defined over a singleton element, then your Hasse diagram will be just a node itself say the element is a only. That means this is a valid Hasse Diagram representing the relation (a, a) . And this relation is reflexive, anti symmetric and transitive and here the element a is both maximal element as well as minimal element.

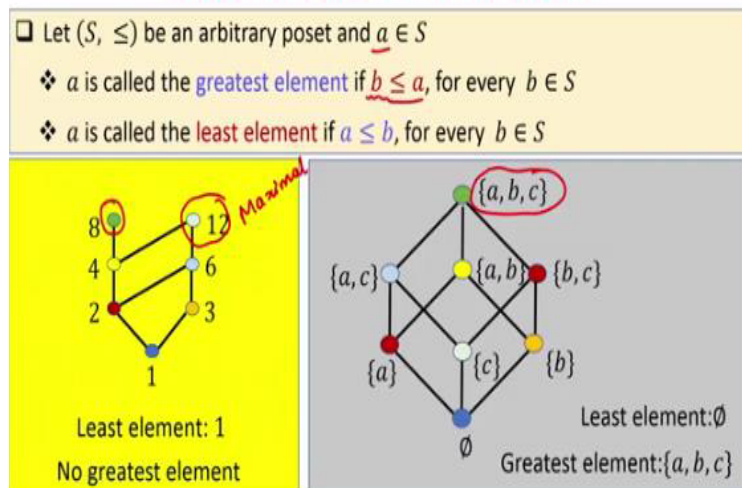
Whereas if your set S has multiple elements and you will have a structure like Hasse Diagram and definitely there will be some element at the lowermost level and some element at the higher most level. So, those elements will be the maximal elements and a minimal. We can prove this thing formally but I am not going into that. Similarly, we can prove that an element of a poset can be both maximal as well as minimal. It is not necessary that the maximal element and minimal element should be different and there can be many maximal element many minimal element.

So, for instance if I consider the equal to relationship $(=)$ over \mathbb{Z} . Then I will have the elements of the form $(1,1), (2,2), (3,3)$ in my relation or the negative ordered pairs of the form $(-1, -1), (-2, -2)$ in my relation. What will be the Hasse diagram look like? The Hasse Diagram will just look like each integer within itself. So, the Hasse diagram will have no edges first of all. Why no edges? Because any element; is related to itself that is all. It is not a related to any other element.

So, the actual directed graph for this equal to relationship will have only self loops. It will have no other edges. And when we construct a Hasse diagram from that diagram, we will remove all the self loops. And as a result we will have a Hasse diagram where no edges will be present in my graph. So, in this graph all the elements are both maximal as well as minimal.

(Refer Slide Time: 30:21)

Greatest and Least Element



Now finally let us define what we call as the greatest element and the least element of a poset. So, if you are given a poset S and with an arbitrary relation less than equal to and if you have an element a then the element a of the set S is called as the greatest element if every element b is related to the element a as per the relation R or the relation less than equal to. In the same way the element a is called as the least element if it is related to every other element b as per your relationship less than equal to.

So, let me demonstrate these two concepts with this example. Here the element 1 will be the least element. Because you have 1 related to 2 you have 1 related to 3 you have 1 related to 4 even though the edge from 1 to 4 is not explicitly there, but as per the notion of transitive as per the definition of Hasse diagram all the transitively implied edges are there in my directed graph of the relation. Then similarly 1 is related to 8 and 1 is related to 6 and 1 is related to 12.

I cannot say that the least element is 2 because 2 is not related to 1 because the implicit direction of the edges are upwards. We do not have downward facing edges as in the Hasse diagram. So, the

least element will be 1, but there is no greatest element. The elements 8 and 12 they are the maximal elements. But none of them is a greatest element, because there is no relationship between 8 and 12 and 12 and 8, they are incomparable elements here. So, I will have maximum elements, but that is not it is not necessary that I should have the greatest element present in my poset.


If I take this poset then here the least element is ϕ because ϕ is related to all other subsets as per the subset relationship. And the greatest element will be the subset $\{a, b, c\}$ because all other elements in this poset are related to this element $\{a, b, c\}$. So, if at all the greatest element exists in my poset, it will be unique but this is not necessary that a greatest element does exist in my poset. Similarly if at all these elements exist in my poset, it will be unique. But it is not necessary that every poset should have a least element.

(Refer Slide Time: 33:06)

Topological Sorting

☐ Input: A set of tasks S and a dependency relation R where aRb , provided task b can start only after finishing task a

☐ Output: To get a schedule for tasks in S



☐ Possible outputs:

❖ 1, 2, 3, 4, 5, 6

❖ 1, 3, 2, 4, 6, 5

❖ 1, 2, 3, 4, 6, 5

❖ 1, 3, 2, 4, 5, 6

☐ Each output can be viewed as a total ordering \leq over S , compatible with R :

If $(a, b) \in R \Rightarrow a \leq b$

(Elements related as per R are still related as per \leq)

Using all the concepts that we have discussed in, now we will now do a very interesting exercise here, which we call as topological sorting. So, in this topological sorting, you are given a set of tasks which is denoted by S and you are also given a dependency relationship R defined over the task in the set S and task a is related to task b provided b can start only after finishing the task a .

And what we want here is we want to get a schedule according to which we should finish the tasks in given in the set S . That means we have to decide which task to finish first and then which start to finish next and so on provided I have the dependency among the tasks given in the form of this

relationship R . So, it is easy to see that the dependency relationship here is a partial ordering which can be described by a Hasse diagram. So, I am taking here a collection of task 1, 2, 3, 4, 5, 6 and a dependency relationship is given like this and what I want here is a schedule for scheduling the various task in the set S .

So, there can be multiples schedules possible. So, I have listed down four of them. I can finish the task one first and then I can finish the task 2 and then the task 3 then the task 4 then the task 5 then the task 6 that is one way of satisfying the requirement. Because once I am done with task 1 the dependency is over one and now I can freely choose either to do task, 2 or task 3. So, if I decide to task if I finish to, decide task 2, then next I can decide to either finish task 3 or I can decide to finish task 4. So, depending upon in what sequence I follow I choose the next task to complete that will give me for different possible schedules. So, it is not the case that there is only one possible schedule here. There can be multiple possible shapes here.

Now each of the possible outputs that I have stated over can be viewed as a total ordering over the set S compatible with my relation R . And why total ordering? You see in my original dependency relation, there are incomparable elements. So, for example, neither 2 is less than 3 nor 3 less than equal to 2 because they are not dependent at each other. Both of them depend on 1. As soon as I finish one I can freely decide or I can freely choose either the task 2 or the task 3. There is no dependency between the tasks 2 and 3 that is why it was only a partial ordering.

But if I say that my final schedule is this then in this final schedule is this then in this final scheduling I am saying explicitly that 2 is related to 3. That means I should I have finished 2 and then I have finished 3. So, in some sense this output sequence, which I have obtained here one of the possible output sequence of that I have obtained here can be considered as a possible total ordering on the task 1, 2, 3, 4, 5, 6 compatible with the relation R .

What do I mean by compatible with the relation R ? By compatibility, I mean that if at all there was any dependency between a and b , that means if a was dependent on, if the task a was related to task b as per the dependency relationship. Then in the final sequence which I have obtained in the final total ordering which I have obtained it still the case that a is related to b . That means if in

my Hasse diagram, if I was constrained to start task number 2 only after finishing task number 1, then in the resultant output sequence that constraint should be satisfied, it should not be validated. It is what I mean by compatible with my original relation R.

And you can see that each of the sequences which I have obtained here. In each of the sequences, I am satisfying the constraints which were given with respect to the original relation. None of the dependency which was maintained which was mentioned in my original relation R is violated in any of the output sequences which I have listed down, on any of the schedule which I have listed down.

(Refer Slide Time: 38:06)

Topological Sorting

```

TopSort( $S, R, n$ )
 $k = 1$ 
While  $S \neq \emptyset$ 
     $a_k =$  a minimal element of  $S$ 
     $S = S - \{a_k\}$ 
     $k = k + 1$ 
Output  $\{a_1, \dots, a_n\}$ 
        
```

□ Theorem: If $(b, c) \in R$, then c will appear in the output after b ✓✓

❖ When c is removed as a minimal element, b would have been already removed, otherwise c is not a minimal element at that step

So, the general goal of the topological sorting is the following. You will be given a relation over a set S that relation may or may not be a total ordering. There may be incomparable elements present as per the relation R. What you have to output you have to output now a total ordering over the set S and the total ordering should respect the original relation R. It should be compatible with the original relation R.

That means whichever pair of elements which were related as per the original relation R, they should be still related as per your new ordering. It should not happen that a was related to b in the old ordering but in the new ordering a is not at all related to b. That should not happen. For the incomparable elements you are free to do whatever you want. But the elements which were

comparable as per the original relation R , you have to maintain those that comparability property in the new ordering as well. That is the goal of Topological Sorting.

So, how do we do this? How do we output one such total ordering? So, the algorithm is as follows. We start with k equal to 1 and I will iteratively do the following till my set $S \neq \phi$. As soon $S = \phi$, I will stop my algorithm. So, what I will do is I will start with the minimal element that is there in my set S . And I will list it down; that means I have taken care of that element a_k in my total ordering and I remove that element a_k from the set S .


So, using my set S keeps on getting updated and that is why I start with my original set S here and every time in each iteration, I will be removing the current minimal element of the current set S and I will update the set S . And I will increment k to the next value of k and I do this till $S = \phi$. And once my set is becomes empty, I will list down the elements in the order in which I have removed them in this value. So, let me demonstrate this algorithm with this example here.

(Refer Slide Time: 40:27)

Topological Sorting

❑ Input: A set of tasks S and a dependency relation R where aRb , provided task b can start only after finishing task a

❑ Output: To get a schedule for tasks in S



$S = \{1, 2, 3, 4, 5, 6\}$

1, 2, 4, 6, 3, 5

So, you have $S = \{1, 2, 3, 4, 5, 6\}$. Your $k = 1$. I start with the original S and find out the current minimal element. And in this case, I have only one minimal element namely the element 1. So, I will write down the element 1 to be the first task which should be taken care in my schedule and then I am removing the element 1 from my set S . That means the task 1 is taken care.

So, you can imagine that since task 1 is taken care there is no dependency of other tasks on the task number 1 and hence these two edges vanish from my Hasse diagram. Now, I have to find out the minimal element of the updated S. And I have two possibilities here. Both 2 as well as 3 are the minimal elements for the updated S and it is up to me. I can either choose 2 in my schedule to be the next task or I can choose 3 to be the next task in my schedule.

It is up to be the algorithm does not say that you have to if you have multiple minimal elements which one to choose. So, suppose I decide to take care of task 2. So, $k = k + 1$. So, I am not writing down the values of k here. Since I have taken care of the task 2, that means this task has vanished now. And now I have to choose the next minimal element and my minimal element are 4 as well as 3.

So, in my sequence, I can either put 4 or I can put or I can put 3, it is up to me. So, suppose I choose 4. So, sorry, so I am following the order 1, 2, 4 because I am taking care of 4 here. So, if I take care of 4, I am left with this set S. And now what is the minimal element? I can choose task 6 as well as task 3, because since 4 is also taken care, this edge also vanish. So, my minimal elements are element 6 and element 3. 5 is not minimal because 3 is related to 5. So, 5 is not minimal here.

So, it is up to me whether I put task number 6 or I can put task number 3. So, if I put task number 6 here, then 6 is taken care then I am left with only 2 task here and my minimal element is now only 3. So, I have to take care of the task number 3 and then finally I have to take care of the task number 5. So, that is essence of this algorithm.

In every step you are finding the minimal element, which is there in your updated set S put it in the sequence and remove it from your Hasse diagram. A very simple algorithm. So now we have to prove that the resultant output which we obtain from this topological sorting will be compatible with the original relation R. That means if at all the element b was related to element c or the task b was related to task c in the original relation, then even in the new sequence or the ordering that you have output the element c or the task c will appear after the task b. And that is very simple to prove.

The proof follows from the fact that when you were removing the task c from the Hasse diagram and putting it in the sequence, at that time it was the minimal element. Because in each step you only decide or you only choose to remove the minimal element from the updated set or updated Hasse diagram.

So, when it was the turn to remove the element c , at that point of time the element c was the minimal element in the Hasse diagram. That means at that point the element b would have been already removed from your Hasse diagram. If element b is still present in the Hasse Diagram, you have not removed it yet then you would have removed element b instead of element c because as per your original relationship $b \leq c$. So, no where you would have retained b and removed c , because in your original Hasse Diagram b was occurring on a lower level than c . And that is a very simple fact based on which we can state or throw this term.

So, that brings me to the end of this lecture just to summarize in this lecture we introduce the notion of partial ordering. A partial ordering is a relation, which is reflexive, anti symmetric and transitive. We introduce the notion of total ordering, Hasse diagram and we also saw the algorithm for topological sorting. Thank you!