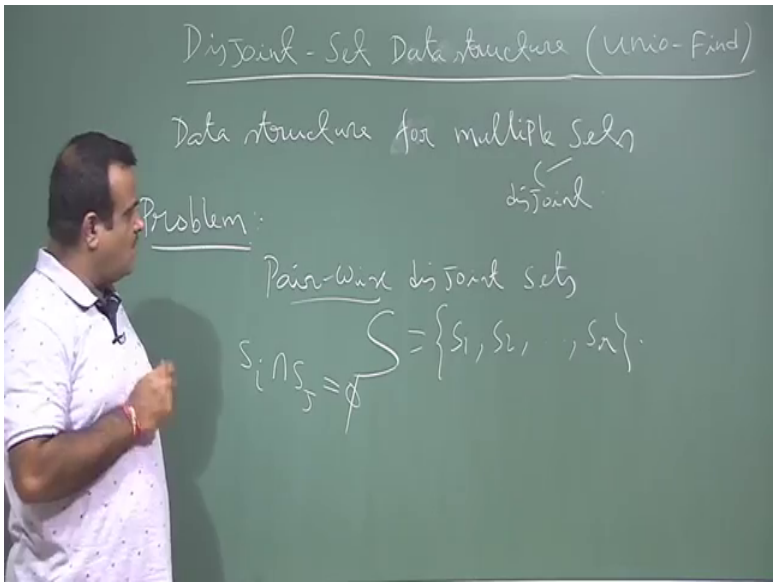
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 51
Disjoint Set Data Structure

We will talk about multiple set data structure like. So, we have seen the data structure for dynamics set, like a set is where the insertion and deletion is going on. So, you have seen hew, we have seen balance tree, red black tree. So, these are the data structure we have seen for a dynamic set. Now we want to look at the data structure for multiple set.

(Refer Slide Time: 00:48)

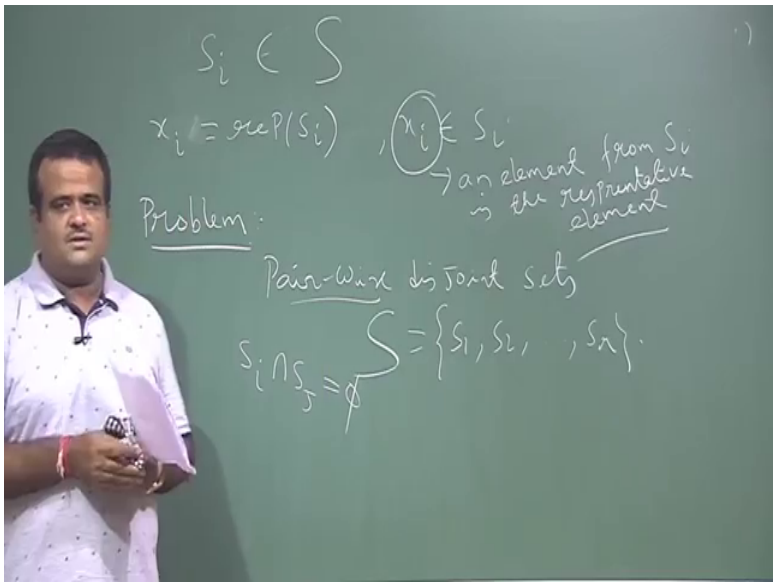


Data structure for multiple set not a single set, and these sets are basically disjoint ok. So, basically we have dynamic sets of disjoint sets. So, the problem is we have a dynamic collection of disjoint sets say- pair-wise disjoint set collection. And this collection we denote by skip test, and these are the set S 1 S 2 say S r, and this is a multiple set collection. And this is a disjoint set pair-wise disjoint means; if you to take any two S i S j they are empty. So, there is no element common to them. So, these sets are and disjoint.

So, this is a dynamic collection. So, any point of time any set can join. So, any two see; what are the operations you want to perform on this set, but this collection is a dynamic collection.

So, we need to have a data structure for this collection of these disjoint sets where we want to support few operations and what are those operations. So, we want to support few operations like. So, before that each sets.

(Refer Slide Time: 02:53)

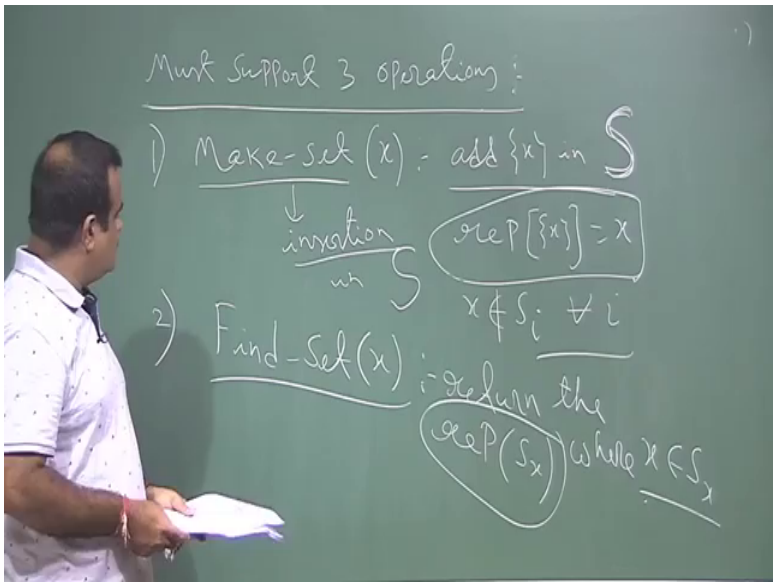


So, suppose S i is a set each set from this script S each set will have a representative element. So, a element from a set we can choose and that will be the; sort of this is the representing of this set. So, this is referred as re P of S i. So, re P of S i is an element in say x i; x i will be the re P of S i where x i belongs to S i. So, we have to choose an element from this is; I mean we can choose any element to be a representative element. So, we have to choose a element from S i which will represent that set. So, that is called representative element of that set. An element from S i is is the representative element.

That means this element is representing the set. So, these we denote by re P of S i. And this element is distinguish element. So we can, I mean this element we should able to tell what is the representative element of this, this is the distinguish element from this set. This is called representative element.

So, given this sets dynamic sets of sets along with the representative element we would like to perform few operations. So, we would like to perform few operations like three operations.

(Refer Slide Time: 04:51)



So, this must support three operations; what are those? So, first operation is make set. So, we can have given a element x a singleton set, so you have to make it a set which is singleton set. So, we have to add this singleton set in skip test; skip test is the set of all set of the; that is the collection dynamics collection. So, there we have to insert, so this is sort of make set is sort of insertion. We are creating a set, so this is kind of insertion operation in this script test. This is sort of what is called a create we are creating a set. So, once we create set then have to insert this into that dynamic collection of sets. So, this is a singleton set we are going to add these two S and then we have to choose the representative element, because each set is having a representative element. Now these set is a singleson set so it has only one element.

So obviously, re P of this set is basically x this element itself, because this is having only one element and that is x. So, that is that is the operation we want to do. And x is not belongs to any other set, because this is a disjoint collection. So, x should not be a member of any other set, because this is a disjoint collection of sets. So, this is the one operation you want to perform, this is the insertion operation or make set operation or create set operation.

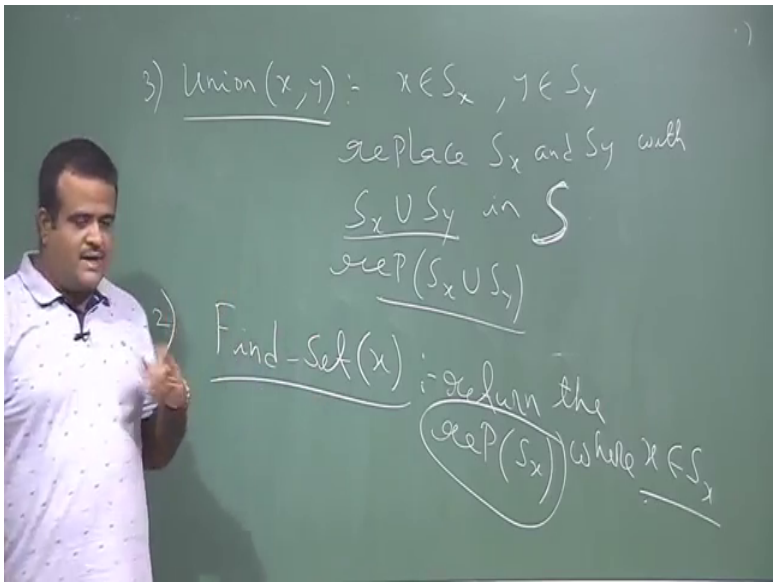
Now we want to perform what is called find set operation. So, find set operation means we have given a element x, so x must belongs to some of the set, we know that set. So, we want to return that set, we want to say x belongs to this set. So, in order to return the

set we just want to return the representative element of the set. We want to say- ok x is in x belongs to the set whose representative element is this. So, that is the find set operation we want to find the set where x is belonging.

So, for that this operation must give us the representative element of the set. So, this is basically returning the return the re P of S x where x belongs to S x. So, you must return the representative element, we must get the representative element of the set. So, basically we need to find the set, this is find set. We need to find the set. So, in order to return the set we can return the representative element, because representative element is representing the set. So, this is the operation which is called find the set.

And another operation is union. So, we want to merge to set like this is called union operation.

(Refer Slide Time: 09:03)



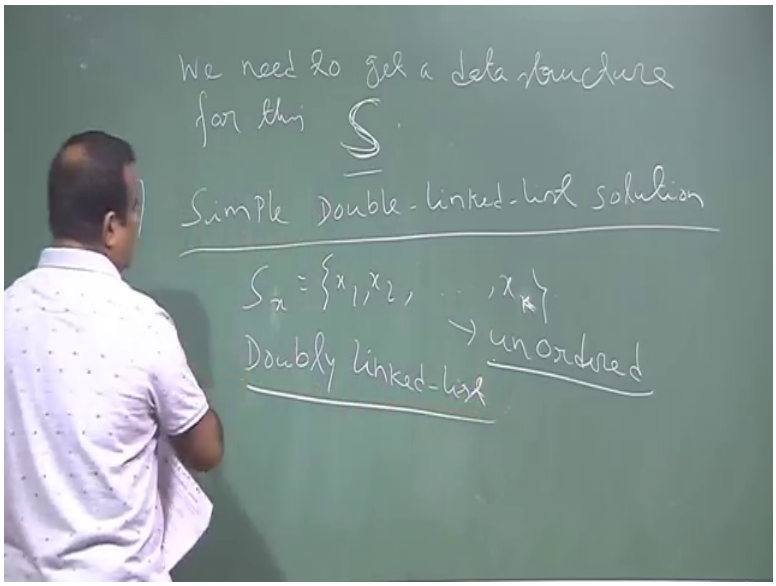
Union of; so we have given two element and this two element will be belongs to two set. So, say x is belonging to S x and y is belonging to S y these two set. Now, we want to merge these two set; that means, we want to do the union of these two set and you want to replace these two set by their union. So we want to, instead to S x S y which want to delete S x S y or you want to add S x union S y. That means, we want to just replace this two set S x S y by their union; replace this two set with S x union operation of two sets, in skip test this is the collection of set yes skip test.

So, where these two elements are distinct element in S x and S y. And once we replace this then you have to find the re P of representative element of S x S y. So, these we have also have to take care in this union operation, you need you need to take care of; we need to decide what will be the representative of this union, not the new set which is the union of these two set ok.

So, this is the problem. So, this is the problem, this is the collection of dynamic collection of disjoint sets where we must able to perform these three operations. One is insertion operation we must create a set, we must create a set that is the make set, then we must find a set- find set means we must given in x, we must able to return the set where x is belonging. So, in order to return the whole set we can return the representative element of that set. And the third operation is union: we should able to merge two sets. And so the problem is to; so we need to get a data structure, we need to think of a data structure for this dynamic collection of sets

So, we need to think of a.

(Refer Slide Time: 11:58)



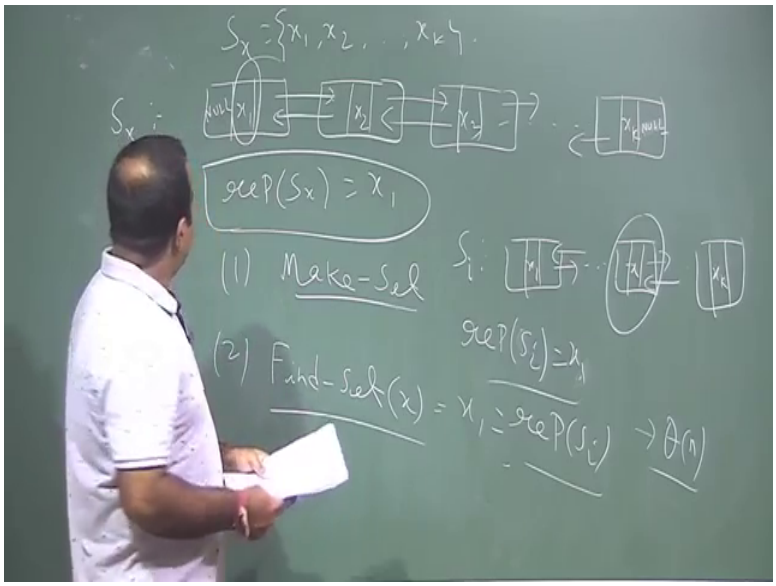
So, we need to get a data structure for this collection of disjoint sets or you should able to perform these operations. So, what data structure you can think of for this collection? So, let us start with simple w connected linked list. So, let us just think of; so this is first suggestion. So, simple linked list w linked list solution. So, we want to store every set in

a linked list. So, you want to have linked list for; I mean given a set suppose this is a S i say x 1 x 2 or this is S x; x 1 x 2 say x k there are k elements there in the set x k, ok.

So, this is set is unordered collection in the set, there is no ordering of the element there is no minimum maximum kind of thing. So, it is a unordered collection. So, set is a unordered; there is no ordering in the set. It is a unordered collection. So, given so we store these element in a w linked list. So, we store this element in a w linked list.

So, how; so basically we will just have a linked list.

(Refer Slide Time: 14:20)



Where we store these elements; so these elements are x 1 x 2. So, this is w linked list, so there is a pointer from this node to this node and there is another pointer from this node to this node. So, this is containing the address of this node. And this field is containing the address of this node; so like this. So, that is the w connected linked list. So, both way it is connected, we can go forward direction as well as we can come we can go to backward direction, ok.

So, this is our S x i mean the set S x. So, you have x 3 like this dot dot dot. Finally, we have x k if there are k element in the set; we have S k and this is null- this is null and this is also null because this is the first element. So, you use these to store the set. So, this is our proposed data structure two for the sets. So, for each of these sets this is S x. So, for S y also you have another w connected linked list, for S j we have another w connected

list. For each set we can have we want to have a linked list; w connected linked list. So, this is ours a data structure we are going to propose.

And we want see how fast we can perform those operation, like make set, how to insert a set. I mean, that means how to create a set, make set, how to find set. So, what is the find operation takes how much time, and then how to do the union? So, those operations you need to; so this is the proposal. So, you want to for this set S i S x. So, we store this in a w connected linked list.

So, now you want to see the operations how we can perform. Like how to perform this set make set operation. And now we need to choose a representative element. So, because this is a data structure you are going to use. So, who is the representative element? So, we want to re P of S x to be the first element. So, this is the first element you are going to choose as a representative element for this set.

So, this is our data structure now you want to see how good data structure is in order to perform this operation like. So, first operation is make set. So, how to make a set we have given in element x which is not there in any other element, because this set has to be disjoint. So, make set means we just create a node with x that is it; this is null this is null. And the re P of this is the first element; so this is make set. So, how much time it will take? This will take theta 1 time. So, make set will take theta 1 time.

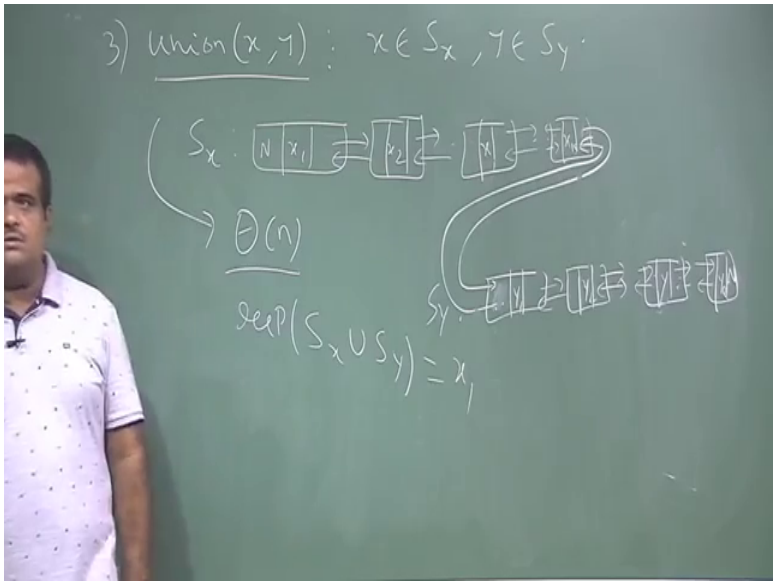
Now, how to find a find set? So, we want to find, we want to return a set where x is belonging. That means, we know that S i. So, this is a S i is containing x. So, this is the set say x 1, so this is w connected and we have dot dot dot we have x over here then we have this is x k. So, we know this pointer, we know this is representative of this is this S i is basically the first node of this x 1. Now to find the set we have given a element and that element is say belongs to x i, we do not need to find the position of x i that is given; that information is given somehow we have a pointer to reach there. So, that we are not finding, so that information has giving somehow we can reach their directly, ok.

So, we are here, we know this S i is containing this x. So now, we need to get the representative element of this set. Then representative element is the first element to reach to the first element what we do we travel the list backward and we reach to the first element we check whether this is first element or not and this is our x 1, we returned x 1 the first element. And that is the find set operation.

So, we returned the x 1 which is basically re P of S i. So, for that we need to traverse the list from backward way, but we this is w connected list there is no issue with that. So, how much time it will take to do that. It will take say linear time, we do not know where is x. So, we go to the x, may be x is the last element and if there are if the sizes, if the total number of element is n then I mean this is a linear time operations.

Then how to do the union? Suppose we want to do the union of two sets. So, that is the third operation you want to perform.

(Refer Slide Time: 21:02)



So, this is the third operation union of two sets. So, x is belonging to S x and y is belonging to say S y. So, you have given two element; that means, we know where these two elements are belonging, they are belonging to 2 sets S x and S y now we into replace this S x S y by their union: S x union S y. So, that is the union operation. So, how to do that? So, this is a S x, so this is x 1, so you have some nodes over here x 2 this is null. So, we have dot dot dot somewhere S x is here, and we have the last element say dot dot dot say x k dot dot dot dot dot. So, this is the S x and we have S y- say we have y 1, we have y 2 dot dot dot. So, somewhere y is here y is one of the element in this set and the last element is say y l. So, this is null, this is null, this is null, this is null.

So, this is the two set S x and S y. Now we want to perform the union operation on these set. So, in order to perform the union operation what we do we just. So, what we do, we just connect these. So, one set say we just connect these and these null. So, we reached to

S x now for after that we want we have to reach to the end of this set. So, then we connect this with this and we know S y we know y. So, you have to reach to the beginning of this and then we have a pointer to this

So, just you are joining this two w connected linked list. So, this is union. So, these set is now we replace this set by S x and S y both by S x union S y this is the merging. So, we are doing this by just connecting the one set with another set with the help of linked list. Just we are adding the pointer on one set to another set. So, this is basically the union and. So, how much time it will take and now what is the representative element of this? So, representative element is first element, our new list have length more it is added the length of y also, but now this is the new the w connected list; and what is the re P of this? Re P of this again the first element which is still remain x 1, if we march this with this then this could be y 1. So, these will see how what is the optimal way to merging this.

So, this is the representative element of this. So, now how much time it is taking to do this union? Now the time taking to do this union is basically because we know the position of this s, but we need to reach to the end of this. Again we know the position of y we need to go to the beginning of this. So, that will take linear time. So, this union will take linear time operations, so because we need to traverse the list to for a S x we need to go to the a end, so for S y you need to go to the beginning in order to merge these two list.

Now, these we want to do it in logarithmic time. So, we want to do it in log n time instead of n you want to perform this also. If I can make set is ok, make set is just a constant time, but other than make set the like find set and union are taking logarithmic time. So, that you want to discuss how we can achieve this by the logarithm. We will think of that this (Refer Time: 25:47) can help us for doing that. And then we will see how we can get the better data structure for this problem. That will do in the next class.

Thank you.