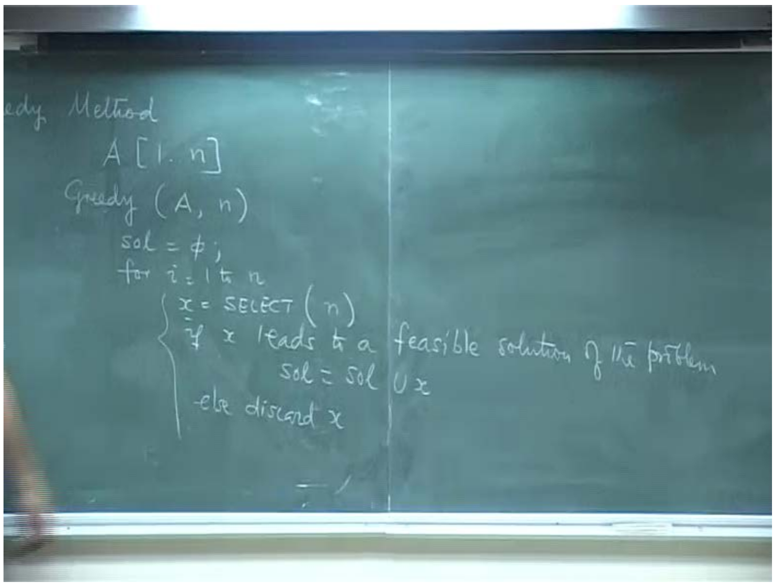
Parallel Algorithms

Prof. Phalguni Gupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture- 3

So, let us discuss some of other types of paradigms as you of (( )) on limited discussing these paradigms with the assumption that you have the basic knowledge of paradigms. And here, we like to like to mention about the tangent problem we handle on the different types of paradigms.

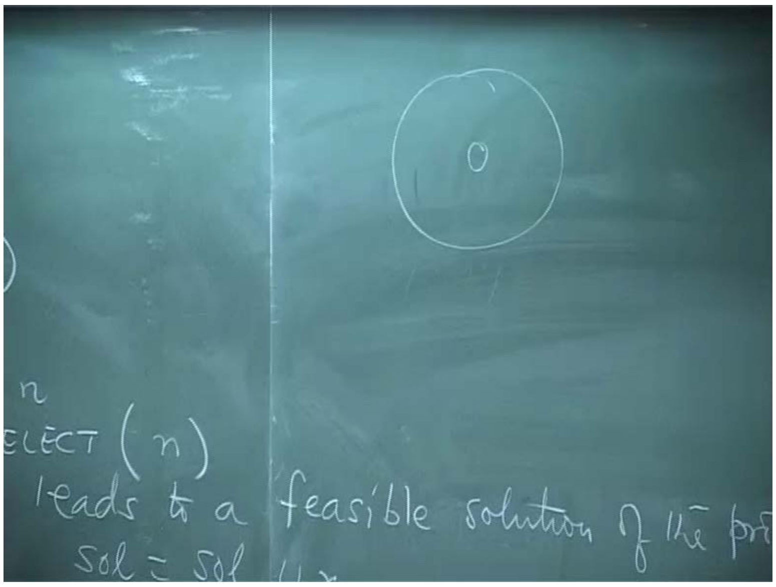
(Refer Slide Time: 00:47)



```
edy Method
    A [1. n]
Greedy (A, n)
    sol = $\phi$;
    for i = 1 to n
    { x = SELECT (n)
      if x leads to a feasible solution of the problem
            sol = sol $\cup$ x
      else discard x
```

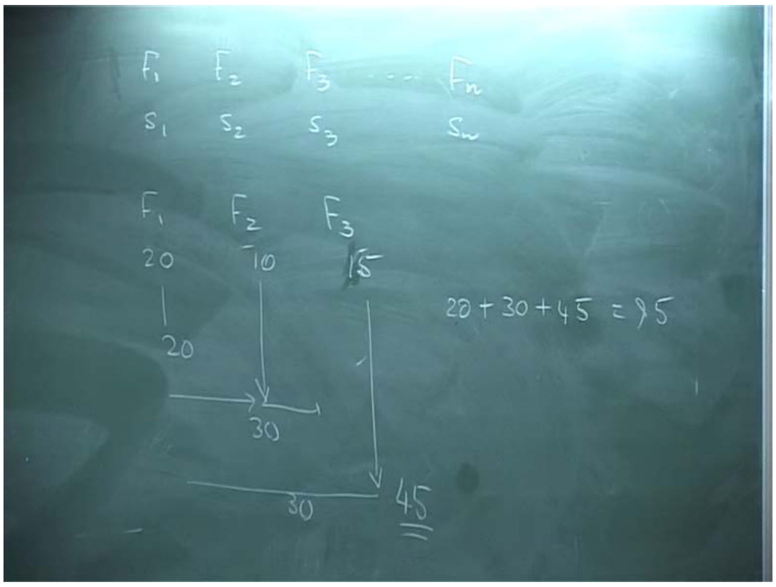
So the second paradigm what would like to discuss today is greedy method. Now here let us assure that here N possible, there are n possible entity, and you want find out the best solution are using the form this n n p k. Now you see you wanted to be it is a, now you have to see select 1 by 1, so that you get the best possible solution for your problem. So, initially you have say if I write the algorithm greedy, genetic algorithm for greedy and A and n parameters, then we will be write it that your solution initially is phi, which is null solution. And then you have to select one after another so for i equals to 1 to n, 1 to n, and then x equals to select 1 parameter out of n and you check you check with a by select x you that solution this will lead to a feasible solution on not. If x leads to feasible solution to a feasible solution of your problem solution of the problem then you update your solution by solution u will in x otherwise, you discard x otherwise discard else discard x. That means that this x will not lead to the feasible solution this is the simple study we follow we follow under the greedy strategy.

(Refer Slide Time: 03:30)



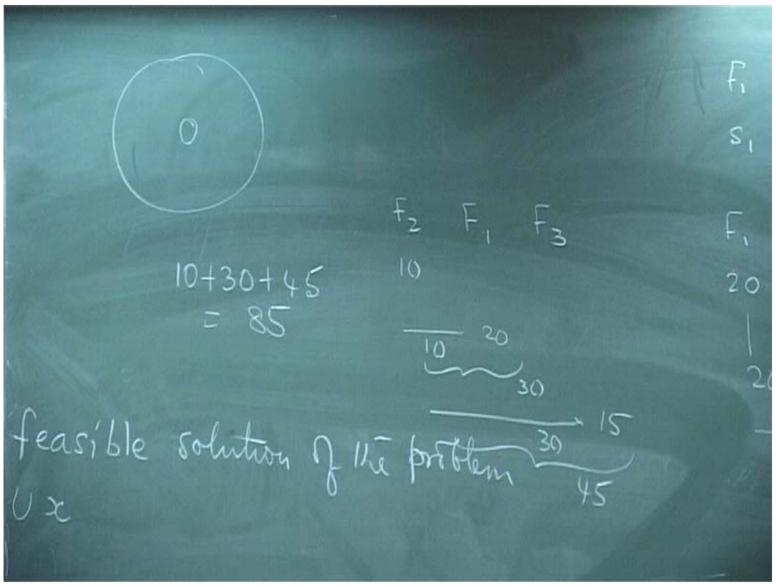
Now one example is there is I do not know whether you have seen the tests magnitude test these are large twice and is to store the data of phi into this magnitude test.

(Refer Slide Time: 03:48)



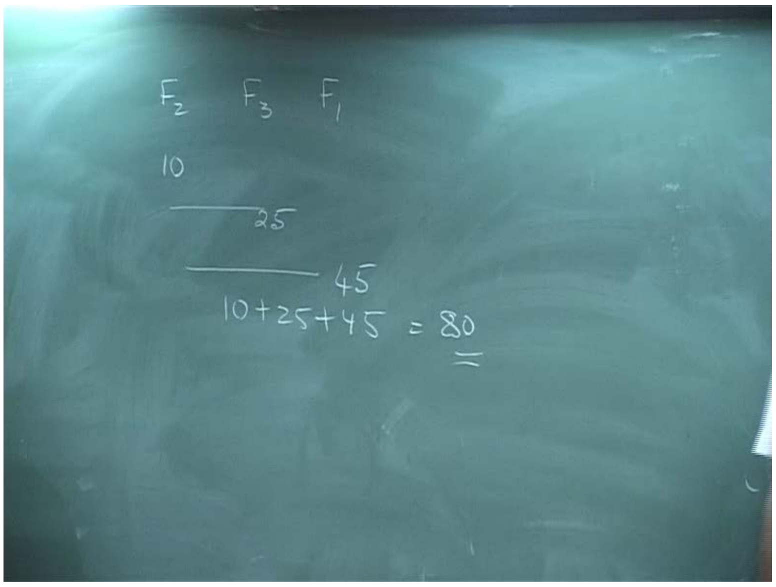
And suppose you have the files F 1, F 2, F 3 and F n and files and the cycle of this phi is S 1,S 2,S 3 and S n. Now, let know store this phi the magnetic in such a way that it takes the minimum amount of phi so suppose I have the 3 files F 1, F 2, F 3 and the size of the phi is say 20,10 and say 5, or 20,10 and that 15. So, if I starts saving that first phi if F 1 then F 2 then F 3 then the total number of decode movements in that magnitude is that first time it will be saving through F phi. F 1 of size 20 records and then it will rewind that so second time if I strokes F 2. Say first to move up to this and then it will write again 10. So, here you need that 30 record movements and third type so it will do not worry and then 15. So, you need 45 record movements the total number of record movement becomes 20 plus, 30 plus, 45 which is the 95.

(Refer Slide Time: 06:16)



Now if I think the other way that know 1 will store first F 2 then, F 1 then F 3. What happens first time 10. Second time is 10 and 10, 20. So, this becomes 30 and the third time this 30 movements plus 15. So, this gives you 45 that mean 10 plus 30 plus 45 this gives you 85 record movements.

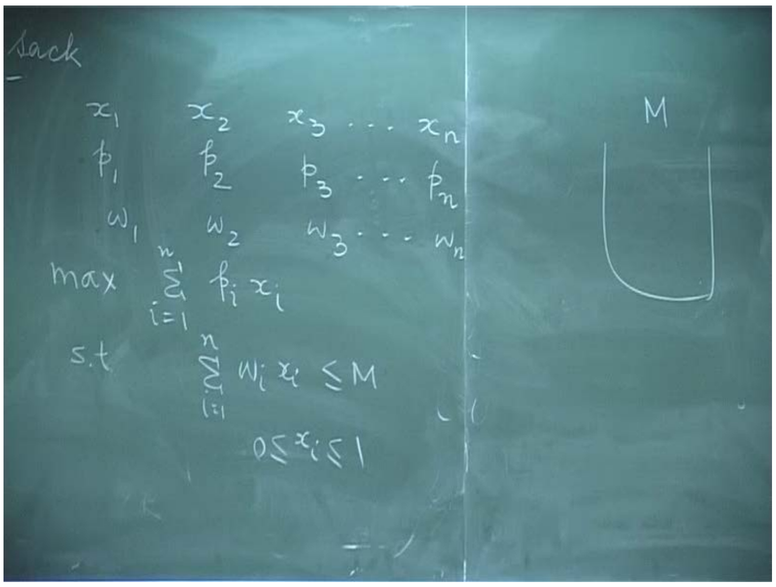
(Refer Slide Time: 06:55)



Now, what happens if I say the files in the form F 2, F 3 and then F 1. First F 2, which is the 10 next times it is 10 plus 15. It is 25 and then next time it is 45. So, it becomes 10 plus 25 plus 45. It becomes 80 and this is the minimum number of rdreco records to be moved to store all this 3 files into the magnitude

So, what is the strategy in follow here that you have n files F 1, F 2, F n and with the size S 1, S 2 S n respectively. The x initially solution is phi and for I equals to 1 to n. Now will be selecting x for from the set of n you will be selecting the phi, which is the smallest 3 files. So S 1 is the solution and is if lifts to F it will lift to the solution. So, you have storing and you solution solve become solve in x 1 x 1 is x which is a smallest phi size and then next time again you take grade the next smallest 1 and you add it and so on. So, this is the one strategy this is the greedy method to solve the problem of storing the files into the magnitude.

(Refer Slide Time: 11:19)

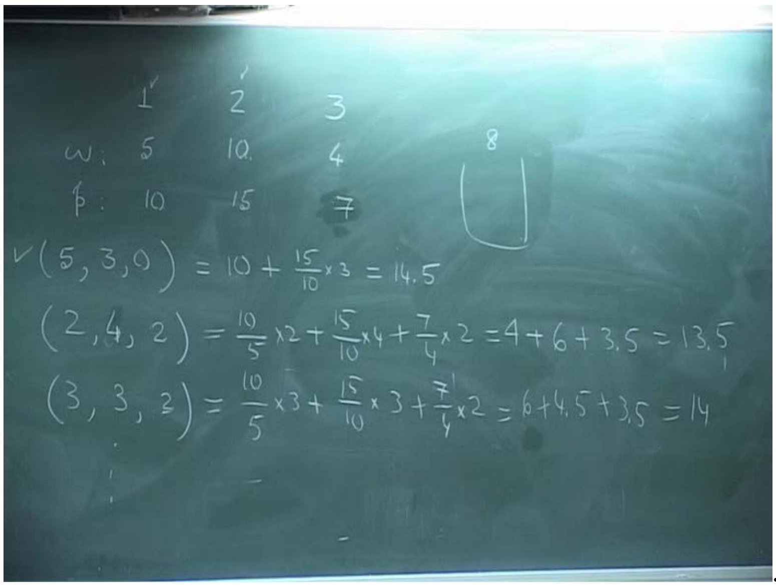

sack
$x_1 \quad x_2 \quad x_3 \ldots x_n$
$p_1 \quad p_2 \quad p_3 \ldots p_n$
$w_1 \quad w_2 \quad w_3 \ldots w_n$
$\max \sum_{i=1}^{n} p_i x_i$
s.t $\sum_{i=1}^{n} w_i x_i \leq M$
$0 \leq x_i \leq 1$
M

Now, on the next problem is well known to you who is knap sack problem. Do know what is knap sack problem right? Is it clear? Yes. Now, let me tell you what is knap sack problem. Let us a show direct you have that n n t ts x 1 x 2 x 3 x n and to select x I you want the profit p i. So, you have the profit p 2, p 3, p n. To select p i you want the profit p i to select x i you want the profit p i and each n t t has started weights say w 1 is the weight at x 2, x 1 weight x 3 and weight n.

Now, there is existing a beam of capacity a bean capacity M. Now from m is to select to select some of xs and put them into this bean whose capacity is M such that you on the maximum profit. So, what is the problem is to select some the xs and put them into the capacity bean of capacity m in such a way that you want the maximum profit right. So, this problem is that if I shall w 3 weight w y weights of item of x i then i are the profit p i and i want to maximize by profit but, with the condition ways that that my bean can take up to the load of M to you need of x.

Now, 1 thing should be clear to you that my bean, if it is not full then I should take some more items from here to get i into the to put into the bean so that i n little more profits. So, I can write that maximize this problem i can write maximize p i x i i is equals to 1. To n such that or subject to the condition subject to the condition summation over w i x i i is 1 to n is less than equals to m and x i is lying between 0 and 1 and ways of possibility right? So, what happens the problem is that i have x 1 x, 2 x n are the n t ts p i is w i is the weight attacks x i and p i is the profit attacks the item x i and you a bean of capacity M. You want to select some of the x i such that your bean is full and also you are in the maximum profit. So, the problem can be read if I maximize summation p i x i x i i equals to 0 i equals to 1 to n subject to the condition that summation over w i x i less then equals to n and x i is 0. And 1 what does it mean x is like by 0 and 1 that it is 0. If you do not select if it is 1 you select the whole item whole item means that you select w i o a ts total right, or you can select you can select small amount of small amount of the w i of the x i col x i ith comp1nt where ith i 10

(Refer Slide Time: 16:30)



	1	2	3
w:	5	10	4
p:	10	15	7

8

$$(5,3,0) = 10 + \frac{15}{10} \times 3 = 14.5$$

$$(2,4,2) = \frac{10}{5} \times 2 + \frac{15}{10} \times 4 + \frac{7}{4} \times 2 = 4 + 6 + 3.5 = 13.5$$

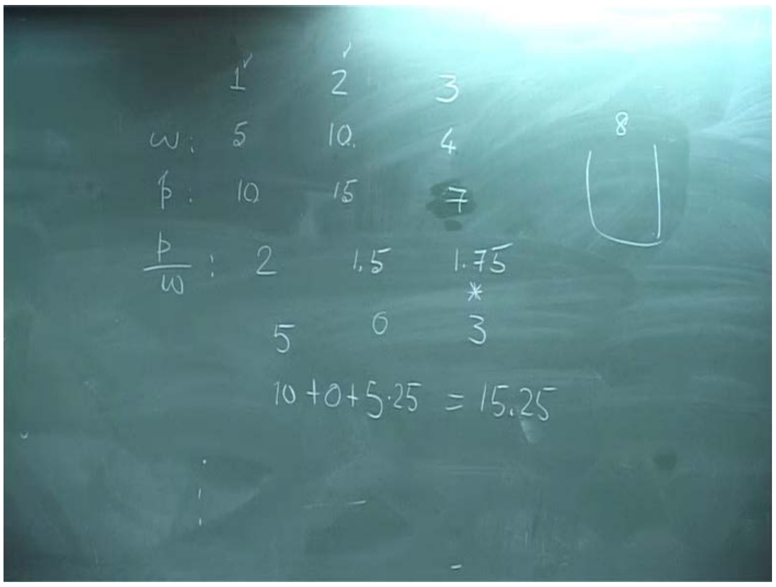
$$(3,3,2) = \frac{10}{5} \times 3 + \frac{15}{10} \times 3 + \frac{7}{4} \times 2 = 6 + 4.5 + 3.5 = 14$$

Now, how to solve this problem? So, let us consider 1 small example suppose I have 3 items item 1 item 2 and item 3 and the item weight is 1 is having the weight 5. This weight is having 10 and this have weight is having say 4 and I earn the profit is say 10. Here it can be 15 and it can be 7 6. No, it can say 7. So, this is your weight and this is your profit. So, there that means that if I shall 5 k unit of item 1 I earn the profit 10 and if I shall 10 unit of item 2 I earn the profit 14 for unit of item 3 I earn the profit of 7.

Now, I want as I have a capacity I have a capacity of say 8. Now, how to select this 8 unit of materials that 1 possible thing that I select item 1 completely and 3, 3 unit from item 2; that means 5 3 and 0 another 1 can be 2 6 and not 6. Say 2 4, 2 may be 1 possibility there of another possibility could be 3, 3 2 and so on. If it is the case that how much I earned the profit or what is the profit amount? So I earned 5. So i earned 10. Then i 3 that means 15 divided by 10 into 3. So, this will gives you so 14 point 5.

Now, in this case 10 divided by 5 into 2 plus 15, divided by 10 into 4 plus 7, divided by 4 into 2. So, this is 1 this 4 plus 1 to 5 6 plus 3 point 5. So, it is 13 point 5. Here 10 by 5 into 3 plus 15 by 10 into 3 plus 7 by 4 into 3 you get 6 plus 4 point 4 plus 3 point 5 is 14 and so on. So, you observe that your this gives the better profit among them but, how to how to select or what should be the best possible way to select weights so that so that your bean is 4 and I earn the maximum profit. So, 1 possibility that because here you observe that we have proceeded that weight as a factor weight is related with by profit so I should obtain what is the per unit profit.

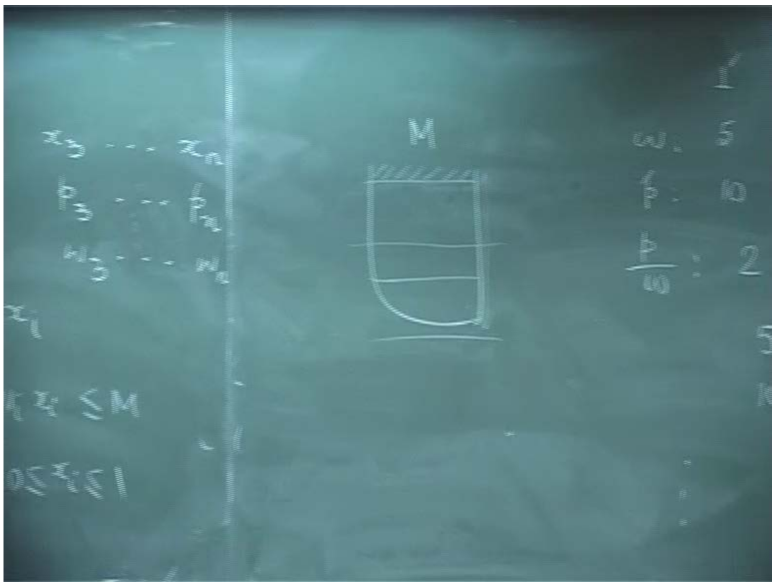
(Refer Slide Time: 18:20).



Now, let us find out the per unit profit that is p by w. So, it is2 it 1 point 5 and this is 1 point 7 5. This is 1 point 7 5 so, if I that means that I must, I must there, I must I put as much as I can from the item 1 into by beans because I am getting final profit is maximum on item 1 and if it is the case if it is the case. So, I will take 5 unit, 5 unit of item 1, then next item is coming this 1. Because this gives the possibility profit is maximum. So, what I should do? I should take 3 unit directly from here 3 is directly from here. So, that and 0 unit from here. So, that you get the maximum profit. So, if it is it is your selection is like that then you are on profit 10 plus 0 plus. So, you earned the profit 15, 20, 5 which is the maximum profit you can earn.

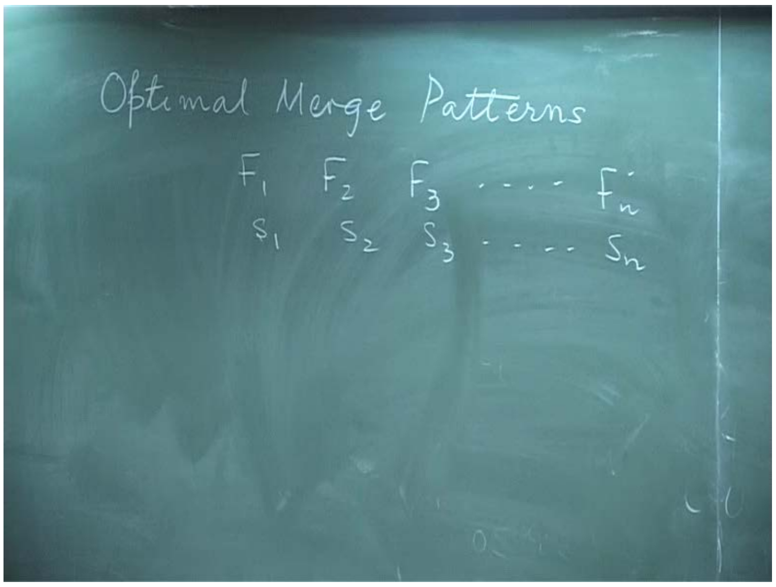
So in terms of in terms of generally in term if I consider the generalize problem that you have have the p 1, p 2, p n profit with reference to item i you have the weight w y then the safest procedure is there you first obtained the per unit profit for each item and after training the per unit for profile in you. Select first the unit having the maximum profit and (( )) as much as you can into the bean.

(Refer Slide Time 19:29)



Now, after putting that maximum profit here you grade the next highest profit 1 and try to put as much as you can into the bean and so on. Finally, you will get some fraction some fraction will be there of the ith 1 to put here. And if I add this you will be getting the maximum profit. So, that is the greedy strategy for finding the knapsack problem finding for solving the knap sack problem.

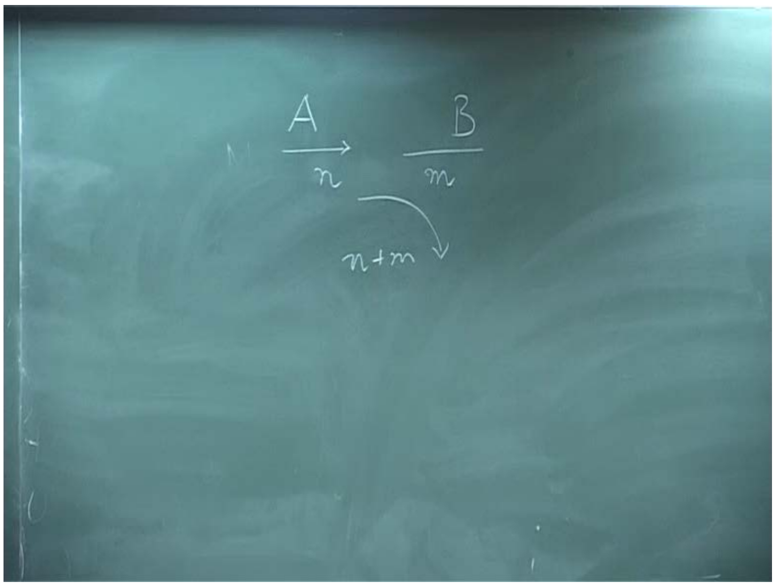
(Refer Slide Time: 20:27)



Now, let us consider another interesting problem that is optimal merge patterns. Now what is the problem here problem is that suppose I have files F1, F2, F3, Fn and the size is S1, S2, S3 Sn. So, you have n files and and and phi i has the size or has the records number s i.

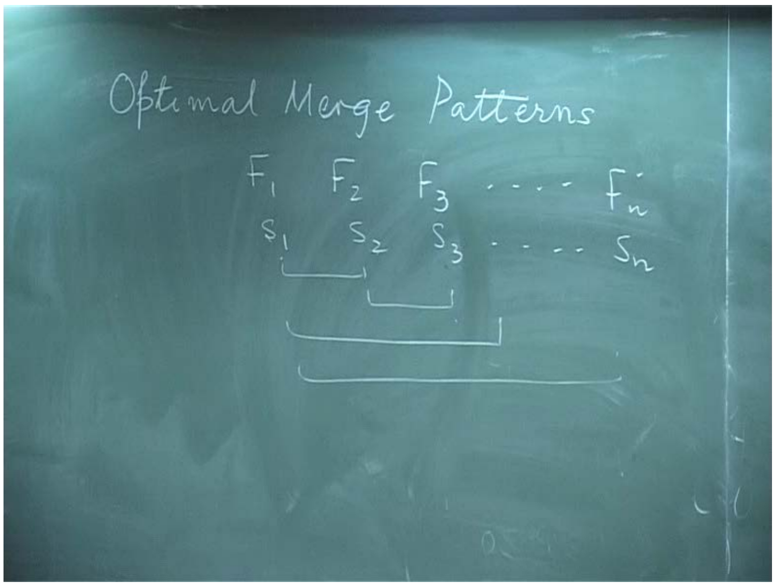
Now, the problem is that we want to merge all this files and to make into 1 phi. Do you observed in the last class we discussed about the merging of 2 files, then there is a need of data to patterns form that you know from 1 file to another file. So, if I have a file A and you have another B and you want to merge you this 2 file.

(Refer Slide Time: 20:21)



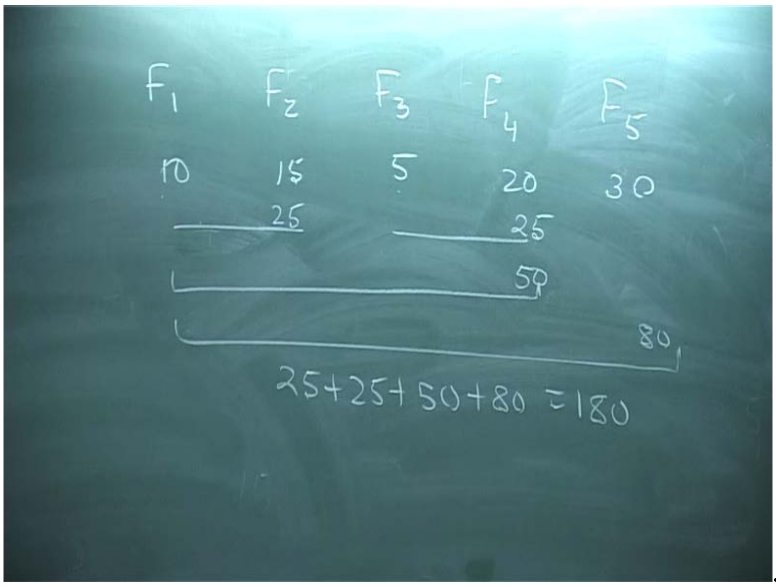
Suppose this is of size n, this is size m and first what you do first you do what you do that this you need n plus m record movements from for merging this 2 files.

(Refer Slide Time: 22:36)



So, for this case also that how I if I merge a F 1 with F 2 then, I need S 1 plus S 2 the record movement and then if I take this 1 and you need again a S 1 plus S 2 plus S 3. There are record book and so on now well that you know our target should be such or our problem should be like that that you merge this files into 1 file in such way that number of records movement is minimum. In minimum there is 1 way could be that you first merge this 2 then you merge item with this 1 merge item with this 1 and then merge item with this 1 and so on right. Another could be that you merge this 2 this 2 and this 2 and this 2 then merge with this 1 this 1 and so on. There is several ways you can merge you can merge this n files.

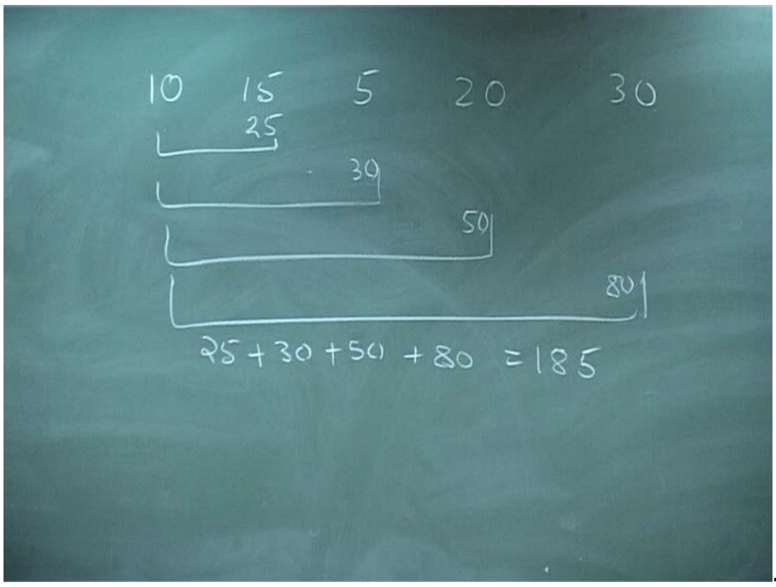
(Refer Slide Time: 23:43)



.

Let you have to understand the problem say let us shown them I have five files F 1, F 2 F 3 F 4 and F 5 and size of the files could be 5 10 20 50 and may be this 1 is 30. So, if I merge for this 1, then this 1 and then merge these 4 files and then these 4 pages. Then let us see a how many record movements will be there to merge this 2 files you need 25 record movements to merge this again 25. Now to merge this 2 you need 50 and to merge this will be you need 80. So, the total number of record movements becomes 25 50 and then 80. So, it is becoming 108 record movement.

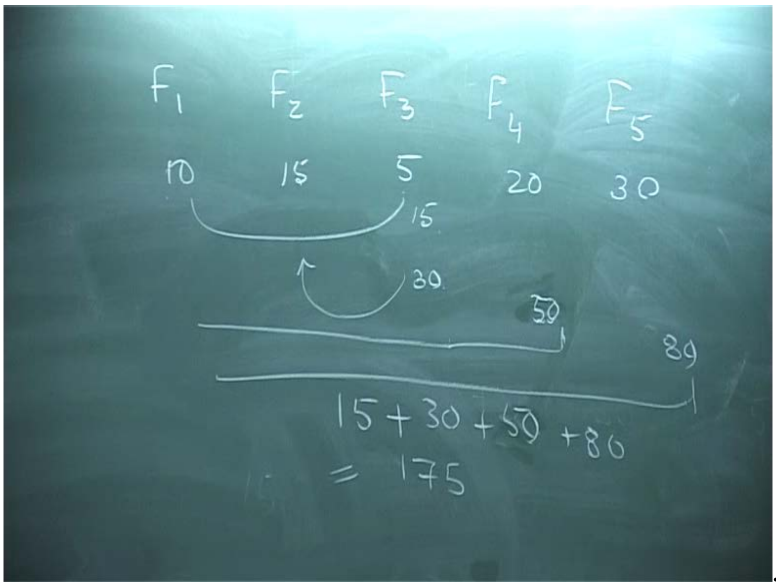
(Refer Slide Time: 24:27)



.

Now let us think about the other way. You have 10 15 5 20 30 and suppose I merge this 1 first then this 1 then this 1 and then finally, this 1 in that case you have 25 record movements here, 30 record movements, here 50 record movements and here 80 record movements. In that case total becomes 25 plus 30 plus 50 plus 80. So, you get 185 record movements is it 195.

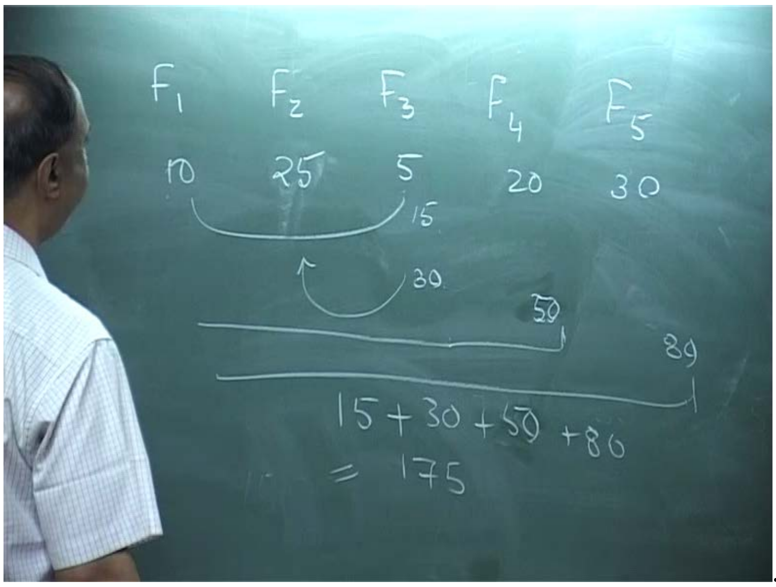
(Refer Slide Time: 25:33)



.

Now what happens file first merge this 2 then that merge item is merge to this 1 then whole thing is merge with this 1 and finally, merge with this in that case here it is 15 here 15 plus 15, 30, here it is 20 and here it is 30. So, number of record movement becomes 15 30 20 30. For 30 this is that 15, then this is 30 and this is our 30 plus 20 50 and this becomes 80 50 and this becomes 80 so this is 40 plus 95 175.

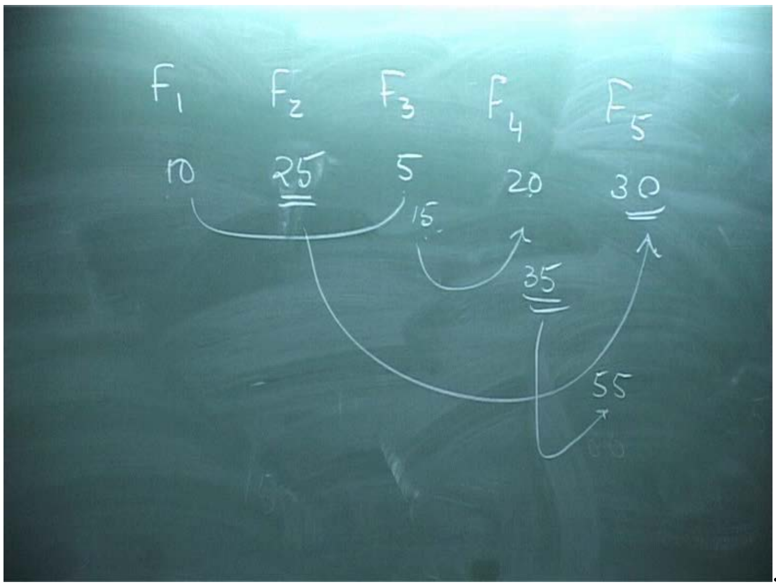
(Refer Slide Time: 25:49)



.

Now can you tell me what is the best way to do it? Yes what you have to do is that you first try to get the file with this smaller size and you try to merge them, then you select again its file with the smallest 1 and you try to merge say for example, suppose this is instead of 15 it is 25 in that case instead of finding this 1 so merge this 1 you.

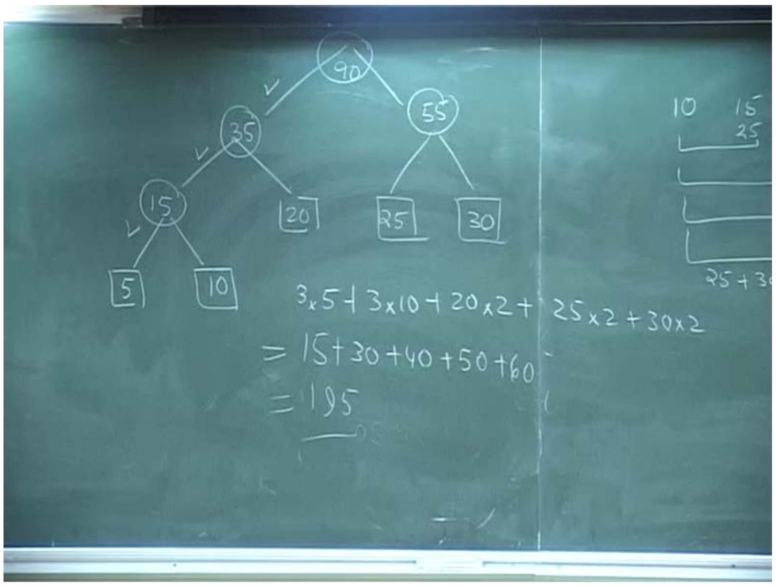
(Refer Slide Time: 26:34)



.

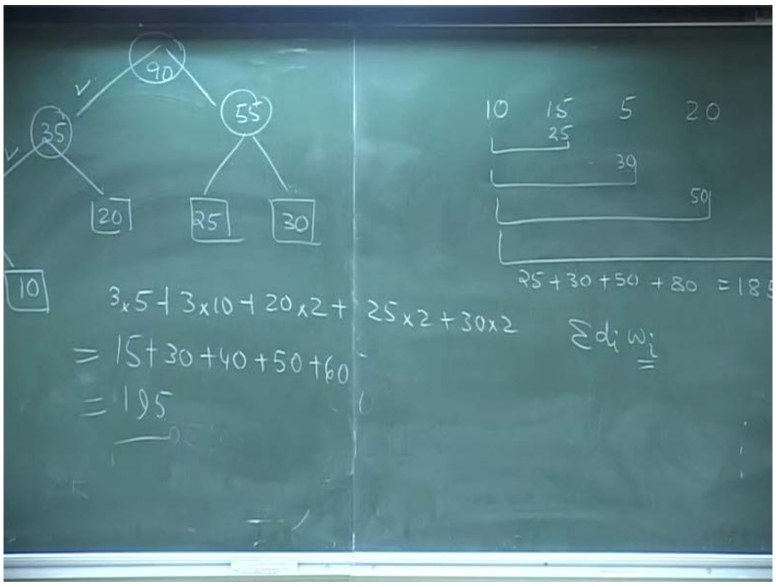
Should have merging this 1 then it becoming is becoming file size becoming 30 five now that this file this file and this file among this 3 that you consist the 2 of smallest 1 with towards the smallest 1. So, this size becomes 55 and then you merge this 1, ok.

(Refer Slide Time: 28:11).



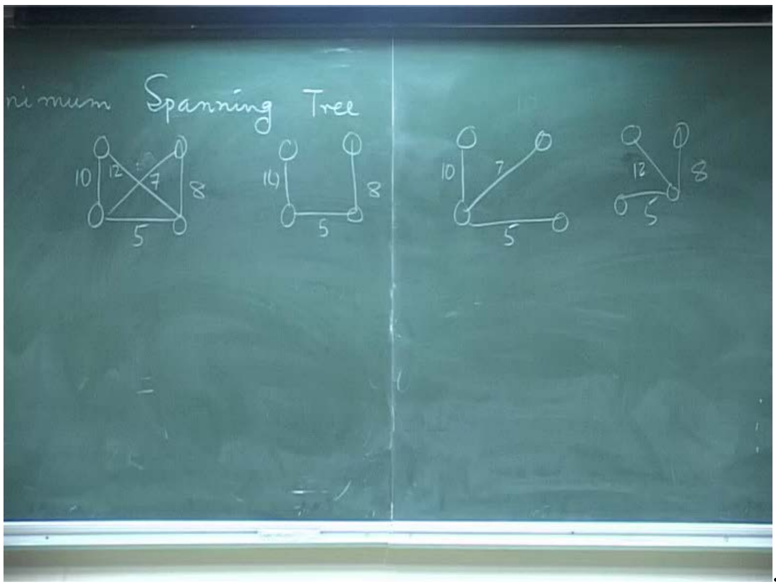
So basically I finally, present in the form of tree. So, what I will do I'll be putting first 5 and I merge with the 10, I get 15 and then I get 20 you get 30 5 and then this side you get 25 and 30 you get 55 you get 90. So, there will number of record movements will be 3 times of file 3 times with this five raise to move up to this so 3 times five plus 3 times of 10 plus 20 into 2 times plus 20 into 2 plus 30 into 2. So, the effect is 15 plus 30 plus 40 plus 50 plus 60. This gives you 150 150 195. So, that is the way you can solve this. So, what you do here the first thing is that you have pick up that 2 files with th smaller in size at the bottom most level and then merge them and you fix up the 2 smallest files again merge them and so on, so, formula becomes d.

(Refer Slide Time: 28:50).



i into wi where wi is the wi is the number of records in the i-th file and bi is the distance of the i-th record from the root.

(Refer Slide Time: 31:14)

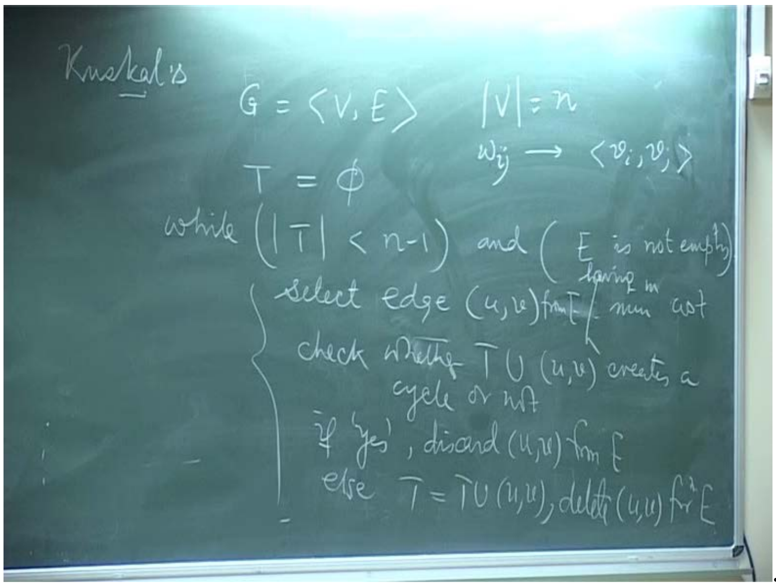


.

Now the next type of problem which we can solve through greedy method is finding the minimum spanning tree. Suppose give me the weighted graph say this is the weighted graph you have and you want obtain the tree from these weighted graph solve the some of the ways minimum and first thing is that if I forget about this, I would looking for the spanning tree then from this tree from this graph you have to find out that the tree with n minus 1 edges and n minus 1 edges that will give you the spanning tree. So, for this graph 1 spanning tree could be this 1. So, this is 10 5 and 8. Another spanning tree could be this 1 which is weight is 10 7 and 5, another spanning tree could be 5 7 and 8 sorry not 7 it is 12 and the cost of the spanning in case the some of this weights. So, the spanning tree having the minimum cost is not a minimum spanning tree.

Now the 2 well known, if you remember correctly there are 2 well known algorithm exist 1 is the fixed algorithm, another 1 is kruskal's algorithm. So, let us discuss kruskal's algorithm and a.

(Refer Slide Time: 36:09)



```
Kruskal's
        $G = \langle V, E \rangle$     $|V| = n$
                            $w_{ij} \rightarrow \langle v_i, v_j \rangle$
        $T = \phi$
while ($|T| < n-1$) and ( E is not empty)
  { select edge (u,v) from E having min cost
    check whether $T \cup (u,v)$ creates a
        cycle or not
      if 'yes', discard (u,v) from E
      else $T = T \cup (u,v)$, delete (u,v) from E
  }
```

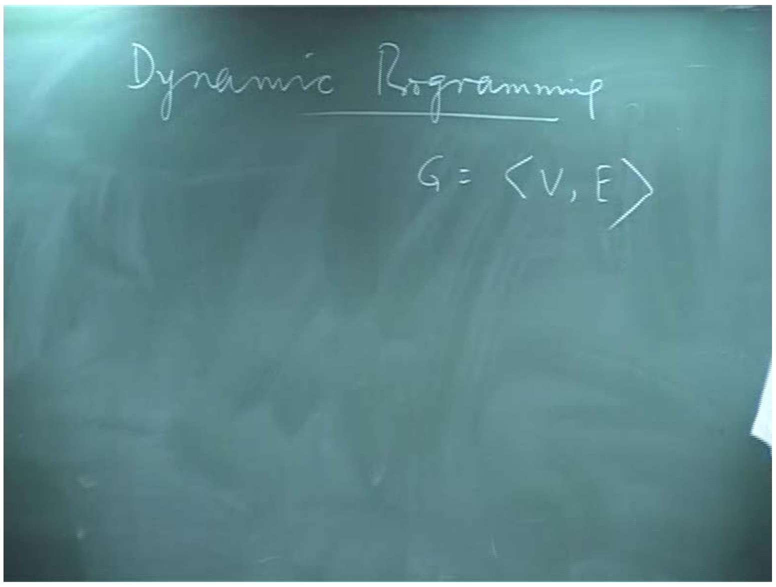
Let see me the graph, we write the set of vertices v and e is the number of edges and you have to assume that number of vertices is n and w i or weight is assigned w i is the weight ascend to the eage between v i and v j. w i j is the weight assigned to v i and v j.

Now since I have to obtain 1 tree, so, the tree becomes tree initial it is not tree and we have to finally, obtain the possible the tree was having number of the vertices varies number of varies as n minus 1. Now, initially this weights here n is the raise in increasing order initially it is a raise increasing order. This with respect to the weights right edges are a raise in the order of in the increasing order of its weight.

Next now the algorithm will be like that, while the number of edges here in the tree is less than n minus 1 and e is not empty t is relevant number of edges in the tree is less than not at n minus 1 and e is not empty. Select edge u v with minimum cost with you select an edge from e having the minimum cost from e, ok.

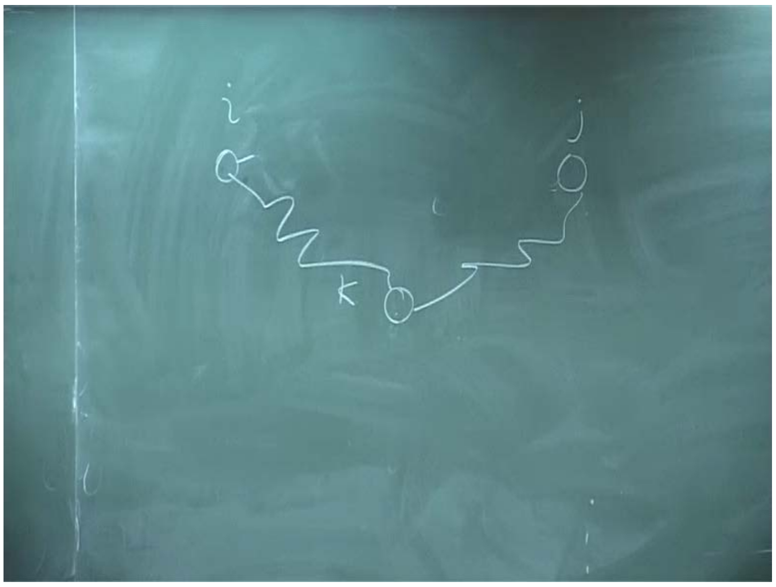
Now you see after selecting this u v you have to say can I put it in my tree? What it means that if I put it in my tree you should not create a cycle, that is the thing you have to see. So, the check whether tree union u v creates a cycle or not, if yes then you cannot considered u for your minimum spanning tree you. Just discard u v form e, else it is not delete you acts as tree union u v and you delete u v form e. So, of that kruskal’s algorithm, so, what it does initial t is 5 while number of edges with less than n minus 1 and is not empty, selecting each u v from e that is must at the minimum cost and then you see if I add this 1 we are in t it should not creates a cycle. If it creates cycle then you will be discard u v from e and otherwise you add it into the tree and then result will be form e. Now after coming out from this while loop you check with the number of edges in tree is n minus 1 if not it is n minus 1. You got the minimum for spanning tree, so, that is the kruskal’s algorithm.

(Refer Slide Time: 37:41).



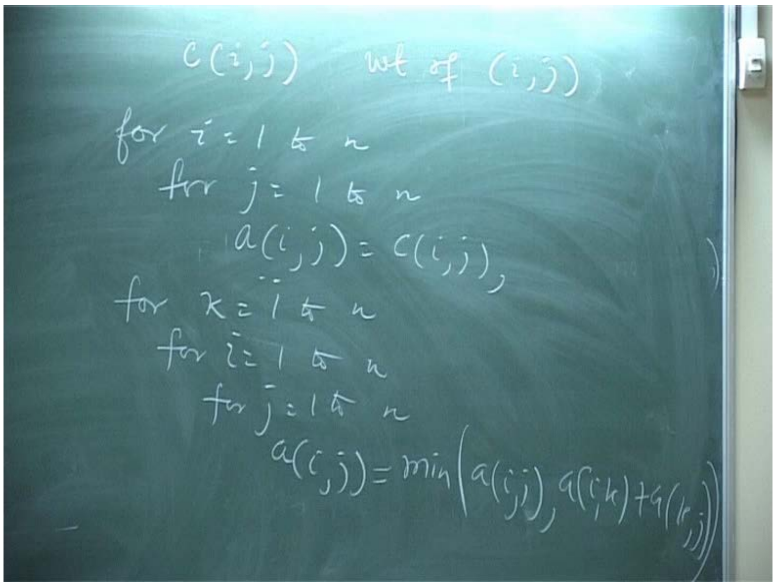
Now next part able is Dynamic program. This is also 1 of the most we known paradigms what we used in different places 1, simple 1 is that finding the all pairs for this first problem and here the strategy is like that suppose you have a graph with a graph g and you have to find the all (( )) what you do that given the cost matrix o a t matrix that you see by inclusion of k-th node whether k-th node ,whether your cost or path becomes shorter or not if it is shorter you consider that is a strategy is feasible that suppose you have i.

(Refer Slide Time: 38:59)



At any instant of j you have the shorted path, you have the paths or solve this paths by a some nodes or by a some vertices lying between 1 to k minus 1. Now by if you have that the k-th stage what we do that we like to introduce the node k and you check the path weight of the path from i to k and k to j. If you find the sum of this weight is less than this the original weight whatever you obtained from i to j then instead of using this path you know, use this path that is the idea. So, dynamically you are updating your shortest this path.

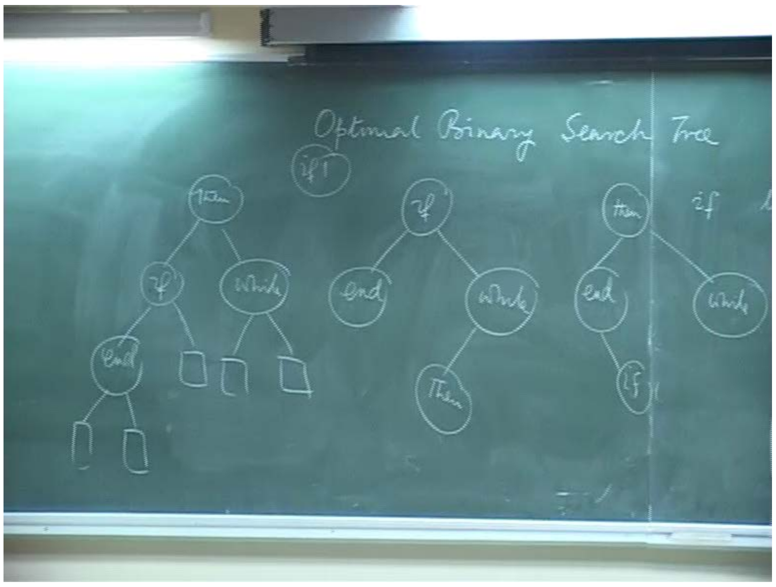
(Refer Slide Time: 40:35)



```
c(i,j)   wt of (i,j)
for i = 1 to n
  for j = 1 to n
    $a(i,j) = c(i,j)$,
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      $a(i,j) = \min(a(i,j), a(i,k)+a(k,j))$
```

Suppose you have written this algorithm, suppose c i c j is the weight of first of the a i j. Now I do not want to talk so for i equal to 1 to n for j is equals to 1 to n. Say a i j is your c i c j and will up by this 1. a i j is a matrix for all path the algorithm is for k equals to 1 to n for i equals to 1 to n, for a is equals to 1 to n a i j in a replace by minimum of a i j and a i k plus a k j what is feasible meaning of that I am consider. Whatever the shortest path I have using I have it will i and j using the vertices 1 2 3 up to k minus 1 and I have the shortest path to perform i to k using the vertices 1 to k minus 1. At last the shortest path using the vertices 1 to k but, in between k and using the vertices 1 to k minus 1 and that will replace by the (( )). This is the 1 thing and similar type of a program we will stable such type of program with (( )) using the binary, using the dynamic programming.

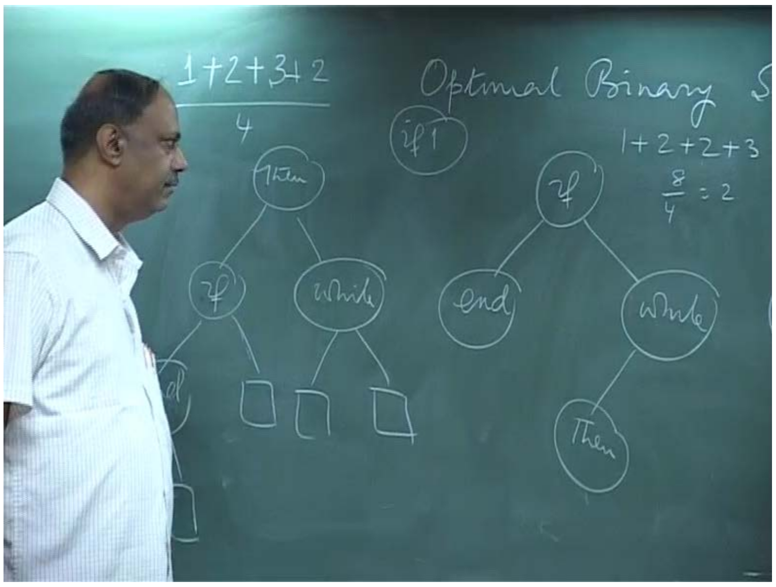
(Refer Slide Time: 44:27).



The another program is where we solve using the binary using the dynamic program is optimal binary search tree. Here what happens the program is that suppose we have the several identifiers say id 1 id 2 and id 3 say id. There are n identifiers and you know this identifiers like to search several time for example, in the case compiler you have started the if then else this are the identifiers and we have retain suppose if 1, now I want to check for the equal reason, I am defined not, so, we will be searching it right say for example, I have if another 1 then another 1 is end another 1 is while right.

Say I have these 4 identifiers; so, I can arrange this or put there in the form of binary tree. Say for example, if I write if is the identify first and this side it will be end, this side it may be while and this side may be else, then another 1 could be you have know I want to put here, then this side it is n, this side it is if this side it is your while. Another way I can put is that can be, say then this side it is if this side it is end, this side it is while. There are several ways, all the way you can have. Now suppose I want to check there whether if exist, if 1 exist so, what I will do? I will search here. Now I will come here, if now will be looking whether there exists anything or not. If it is got then you will tell this is not there. Now similarly, in the case with suppose while exist or not 1 will come here then it has exist and it becomes successful also.

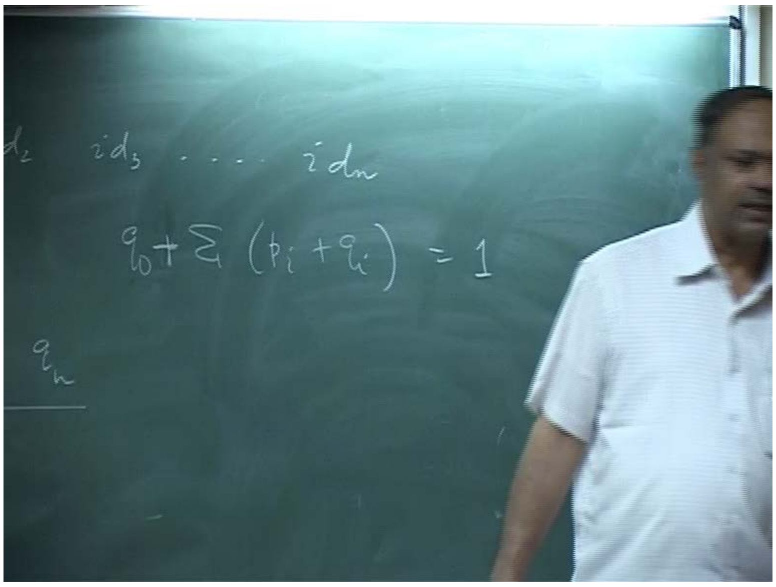
(Refer Slide Time: 46:07).



So, there is question of coming that number of comparison you will be performing. Now find out with that can identify exist or not. Now for this case are if it is a successful search a then very number of comparison required for than is 1 for, if it is 2 comparisons for end, 3 comparisons for while. This 3 comparison so this is 6 plus 6 plus 5 is 11 number divided by 4. So, that is your while number of comparison.

Ok but, for this case it is also 1 2 plus 2 plus 3 and it becomes 5 plus 3 8 divided by 4. So, it becomes 2 so this is that I assume that there are equal lightly than for successful search it becomes 2 point 75 algorithm numbers of comparisons. In this case it is 2 in that case also it will become the 2 point 75 end. So 1 our target should draw at the tree such way that there algorithm number of comparison is minimum. Now the here we assume that all of them are equal lightly and also we assume that that it is only the (( )) but, what happens really this suppose I go for if 1, that it becomes failure 1, right. So, there is not only that we are assuming that equal lightly which is not Correct because, so, I depend upon the program or program i has…

(Refer Slide Time: 48:38).


$d_2 \quad id_3 \quad \ldots \quad id_n$
$q_0 + \Sigma (p_i + q_i) = 1$
q_n

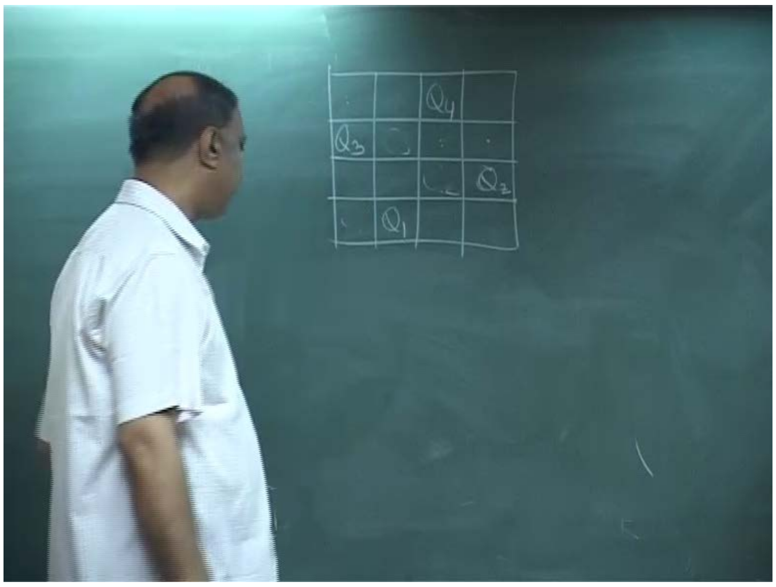
So, 1 part is that the question is coming for the identifier i d1 the probability of successful saw this p i and there will be. You observe that if I have the 4 n identifier than there will be n plus 1 failure such that, if i have 1 item say d 2 and in the case become here to the does not exist. So, it becomes failure search. So, there will be q 0 q 1 q 2 q q i. q 0 q 1 and q n that many they this are the probabilities for unsuccessful search. The face this will lead to this class of identifier this will lead to this i this unsuccessful search leads to this class of items and so on right.

So, since such that summation over p i plus q i plus q 0 is 1 p i is the probability for a i-th identifier successful search and q 1 q 0 q 1 q 2 q n are the probability for unsuccessful searches. Then you have q0 plus summation over p i q of plus q i is equal to 1. Now what is the problem? Now problem is that you have to put the identifier in such way that in such a way that the cost is minimum cost means that if I put i-th identify in the level t than t i then you will be writing as p i and t i and so on.

So, the cost should be minimum. So, this can be solved in using the dynamic programming. Now the next type of probable you have parallel is back techniques and ensuring form I think this also I do not want to discuss merge on this (( )) but, what I want to tell that say 1 problem is that graph coloring problems. So, what you do that you have any tree of the in quotation, you want to color 1 by vertex by 1 color and then will go to the next vertex and put the color another type of color and you check that if by putting that color it fails to satisfy the criteria of graph coloring than if it is then go back and then go off and previous note you just to put different type of color.

So, that is 1 example of back techniques similarly, the case n queen problems that to problem is to put the queen in the chess board that is cost n chess board in such way that no queen attach the Other queen.

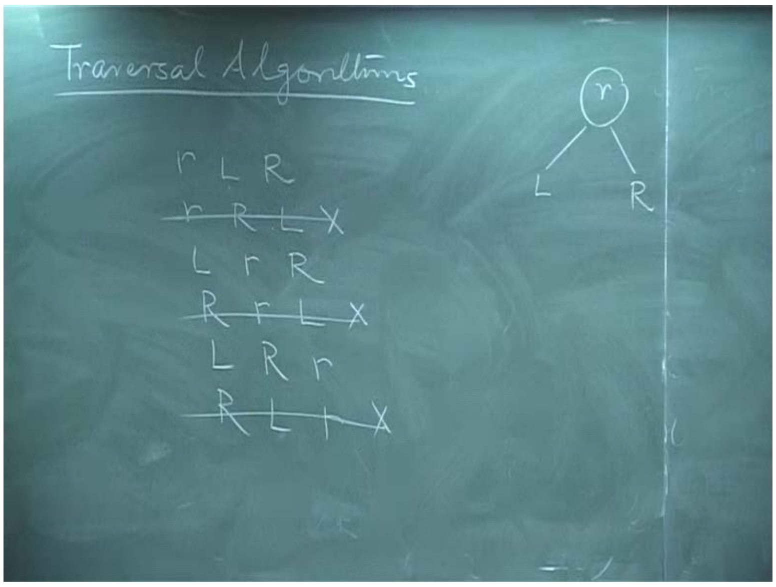
(Refer Slide Time: 52:45)



For example, I have a chess board of cost 4 and I have the 4 queens q 1 q 2 q 3 q 4. So, 2 queens at each other diagonally or if I place them diagonally or what is row algorithm just fitting the saving for us. I would put this incident in such way that they are not on the same diagonal. Not 1 the same row and not on that same column. So, if I put q 1 here q 2 cannot be put here q 2 so I can put q 2 here. Once I put the q 2 here which like put q 3 I cannot put here I cannot put here I cannot put here I cannot cannot I cannot put here I cannot put here I cannot put here I cannot put here I cannot put here. Once I cannot put here q 2 q 3 and so what shall I do with the assume for 2 simplify and under the assume that q i is placed with the ith row. q i is placed in the ith row. So, q 3 should be place here now. Once I know that q 3 position cannot be this 1, cannot be this 1, cannot be this 1, cannot be this 1, what shall I do? I back track. Now I tried to put the position of q 2 so cannot be here. Let us assume that q 3 is here. Now what happens q 3 cannot be here q 3 yes q 3 can be here so, I put q 3 is here right because it is not attacking by q 1 is attacking by q 2 now what happens what shall I put what can I put by q 4.

Cannot put q 4 here q 4 no q 4 no q 4 no q 4 here. Now what shall I do know now you back track cannot put q 3 here so cannot put q 3 here no, now cannot put q 2, q 2 has g1 so, I have to put q 1 independently so I can put q 1 here. Now if I put q 1 here what shall I put p 2? I cannot put here here here, I can put here. Now if it is case when can I put q 3 q 3 I can put here if I put q 3 here q 4 I can put here. So, you observe that you got a solution where they do queen is attacking each other, now there may be several. It is not the unique solution. You may have the defined cost depend way is to put this values for queens in the chess board to get the solution.

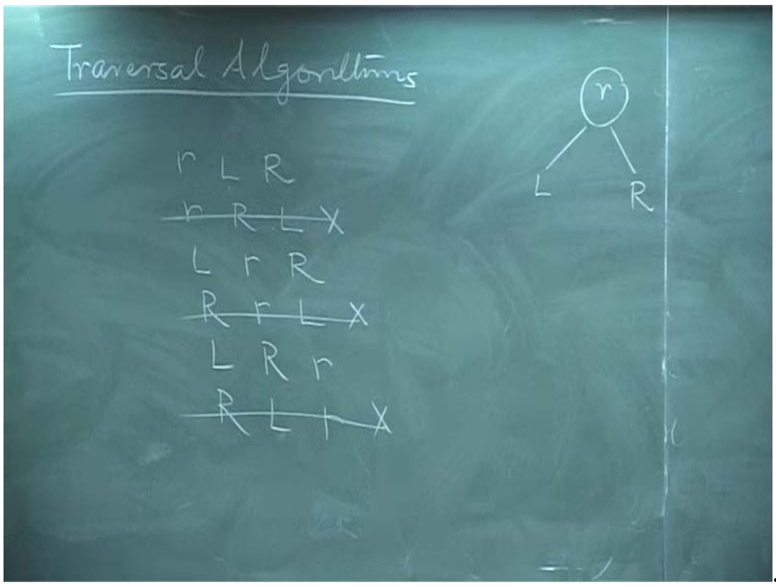
(Refer Slide Time: 55:44)



Get so, this is all about the back tracking. Similarly, you can solve you can find out the branch in found and the other thing from it. 1 thing! Want to discuss is the traversal algorithm. This we need to know for a word which is studied. Now first problem is that suppose I am a binary tree and I want to traverse is binary tree. How you can traverse? Now we tell this is the root and you have the left of tree and you have the right side, ok.

Now you want to traverse is binary tree with nodes. There are several ways you can traverse. 1 way could be I traverse first root than lefts of tree than rights of tree and recovers ably. I go for the left of the tree root left side and so on another 1 could be roots right side tree left side tree another 1 could be left side tree roots right side tree another 1 could be right side tree root left side tree. Another 1 could be left side tree root right side tree root for right side tree left side tree. These are the 6 possible ways you can traverse binary tree. Now see this is the basically the mirror of this saw clearly we do not consider this thing. So basically we have the 3 ways to traverse a binary tree 1 is the root.

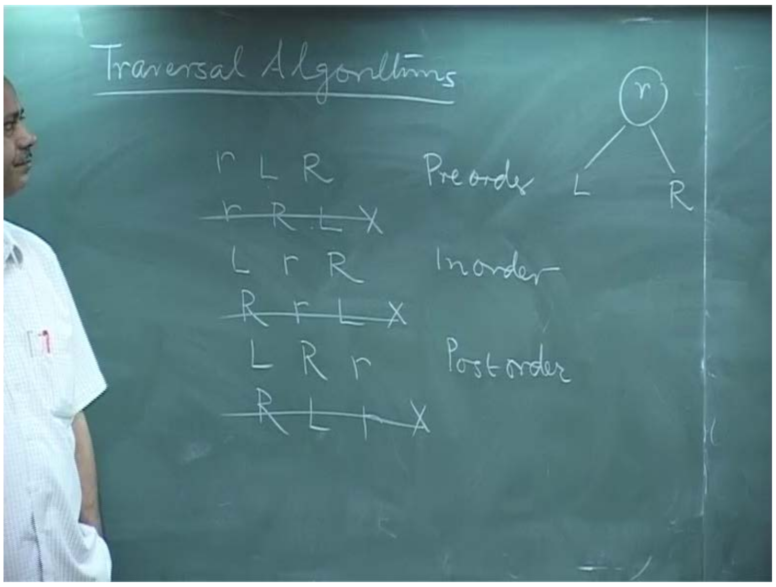
(Refer Slide Time: 55:04)



.

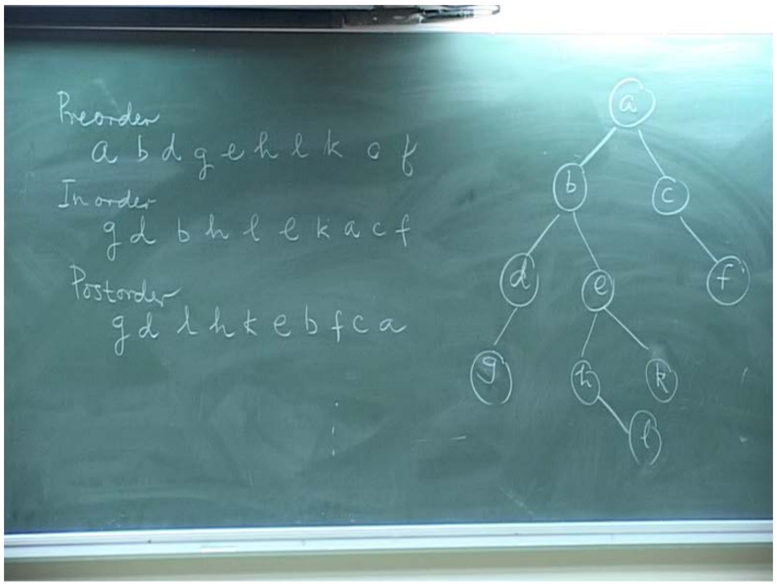
First then right left sub tree then right sub tree. Another 1 is the left sub tree root right sub tree another 1 is left sub tree root right sub tree. Say for example, if I have a b c d e f g h suppose. You have this binary tree now this traversal is known as pre order traversal.

(Refer Slide Time: 56:15)



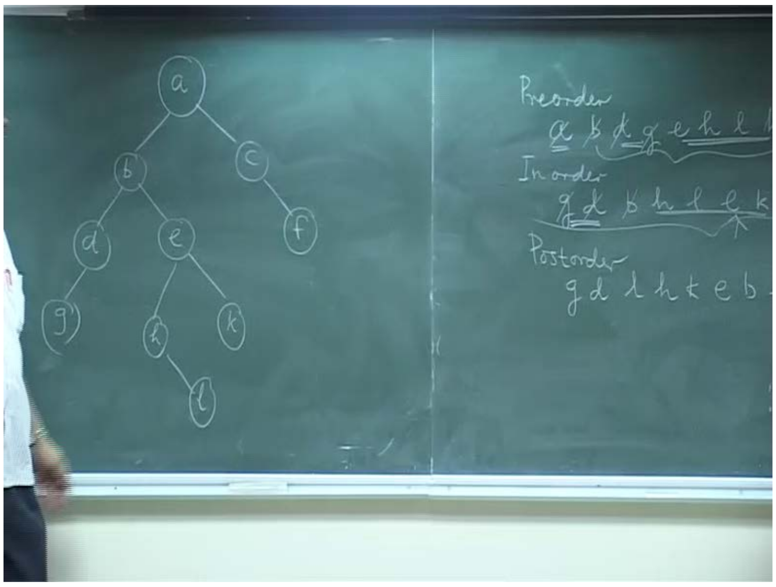
This is known as in order and this is known as post order. So, if I traverse pre order using the preorder traversal then pre order tells the root first the left sub tree then right sub tree now left.

(Refer Slide Time: 58:28)



Sub tree. this 1 so if i traverse then bs ,then left sub tree, b then g then e then h then l then k then c and then f now. If I traverse in order then first is left sub tree then root right sub tree. Now you have come here first is left sub tree then note the right sub tree. Now you have to first left sub tree, then root. So, it becomes that g d b then h l e k a c f and if I traverse post order then left sub tree right sub tree and then root. Now left sub tree right sub tree then root then left sub tree right sub tree. So g d right sub tree left sub tree right sub tree is l h k e b f c a. So, I think this is remember we discuss all this things in your undergraduate curriculum? The 1 thing you remember that if I know 1 of the design the preorder the post order and in order, I can generate the binary tree you need to but, this is of the case if I know. Preorder post order are the not the in order you have to see that. Suppose I know that preorder and in order let us see how we can generate the binary tree.

(Refer Slide Time: 01:01:24)



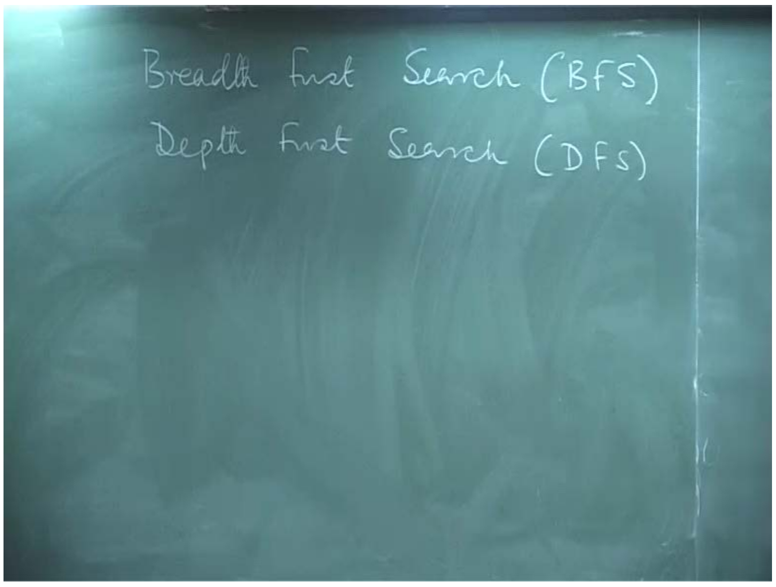
See from the preorder I can get the, who is the root of this node, which is a now from the in order. I can find out what are the elements in the left sub tree are and what are the elements in the right sub tree. So, this gives you that this much is 1 the left sub tree this much is on the right sub tree.

Now from preorder again I can find out who is the root which is b again. From here I can find out this is the left sub tree and this is the right sub tree of d. So, g d is here and this is here now from the d g g d i get d. So, I know the d is the root and g is d is the root and from here you get the g is the left of d. So, it is g now once you know d. So, I have d1. This I have d1 this 1 this 1 this 1. Now I have this part is the root, so, I write here e once I know the e is here h and l is the left sub tree of e h and l so h is the root and l is the right sub tree and e k k is the right of e. So, k then you have c and f c is the root and f is the right.

So, similarly, if I have the preorder and I have if I have the post order in order, I can generate the binary tree uniquely. But, in the case I if I have the preorder and post order I am in problem. I know from the preorder I can get the root of the tree which is here but, after that I do not know which are elements in the left side and right side. So, as the result you cannot go out the conclusion of it.

Now the next 1 is that traversing the graph. If I complete this 1 then you have the preliminary of algorithms particle but, so, then you can go for the parallel algorithm materials. So, there are 2 way we can traverse the graph, 1 is known as breadth first search in short.

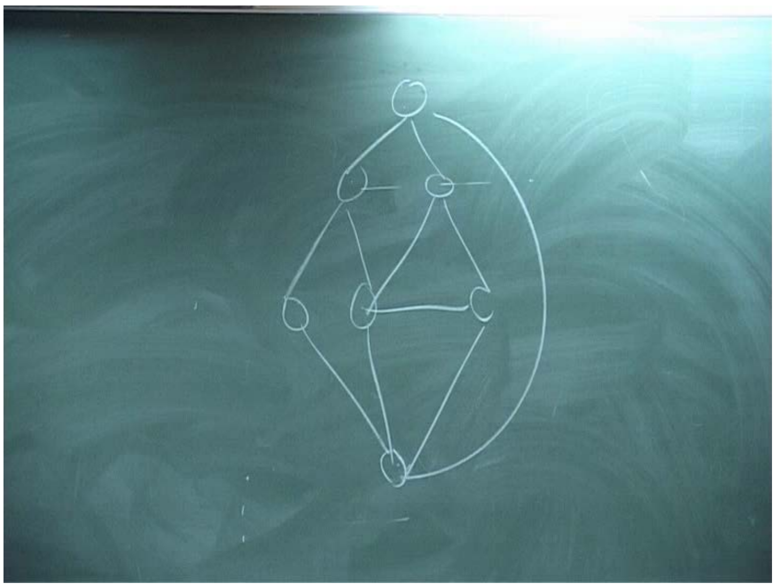
(Refer Slide Time: 01:02:35)



Breadth first Search (BFS)
Depth First Search (DFS)

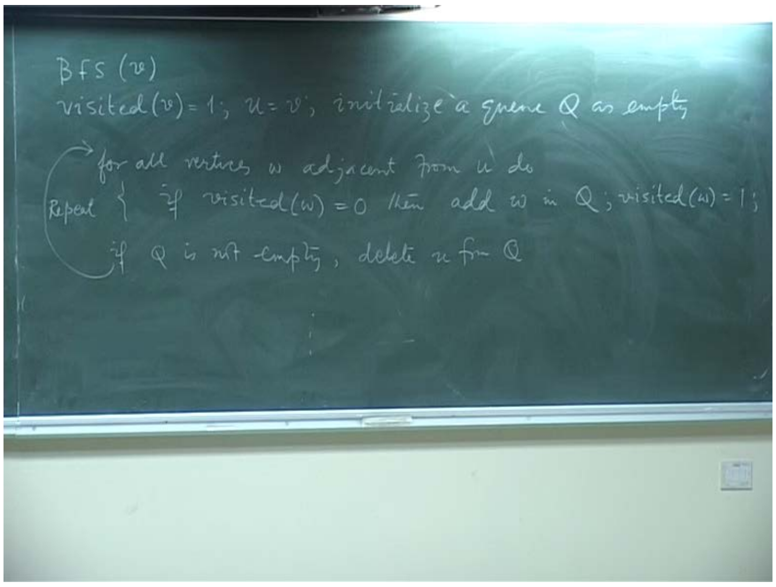
We tell bfs another 1 is known as depth first search in short dfs. So, difference between these 2 is then suppose you have a graph suppose and what I have it does that 1 is that you first go to select this, visit this slope and then you visit all vertices of its set. Then for each adjacent then you go to the next 1 and so on so it is the level wise basically you have search

(Refer Slide Time: 1:02:49)



The other method is that you saw first search is to come to this, then you come to this, come to this, then you finish this, 1 then you finish, 1 that is known as depth first search. Hopefully, you remember those things but if I have to write the algorithm for breadth first search say we can.

(Refer Slide Time: 01:06:10)

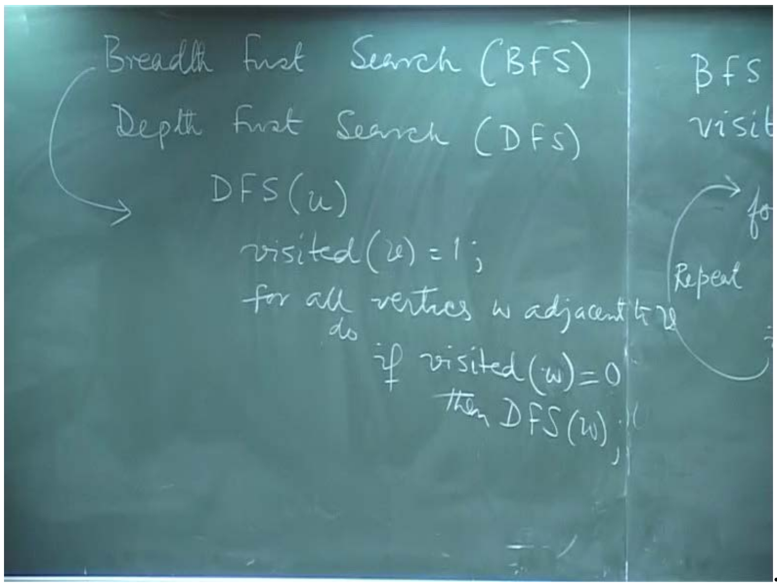


```
BFS (v)
visited(v) = 1; u = v; initialize a queue Q as empty
       for all vertices w adjacent from u do
Repeat {     if visited(w) = 0 then add w in Q; visited(w) = 1;
       if Q is not empty, delete u from Q
```

Write bfs and v is the starting node then visited v is 1 and I write u equals to v now for all vertices in also initialize q is empty, for all vertices u adjacent from u do the following. If visited w is 0 that means not here visited then add w in q and visited w in set to 0 set to 1 set to 1. So, this comes under follow and if q is not empty not empty delete u from q and repeats and repeat this ok.

So, what it does basically for all vertices w and adjacent from u do the following if visited w. w is not at visited then we visit that 1 and you add w into the q and you repeat. This if for all adjacent vertices that is the same level takes and if find that q is not empty you take the 1 delete 1 element from q from the top and repeat this process because you have to find out w and so on. But, in the case of that depth first in the case of depth first the (( )) is little you go into the depth of the graph and the algorithm becomes different.

(Refer Slide Time: 01:08:03)


Breadth First Search (BFS)
Depth First Search (DFS)

```
DFS(u)
    visited(v) = 1;
    for all vertices w adjacent to v
    do
        if visited(w) = 0
            then DFS(w);
```

BFS
visit
Repeat

.

Little different dfs u and visited algorithm becomes like that v is equal to 1 and for all vertices w adjacent to u adjacent to v if you have consider v do the following if visited be visited w is equals to 0. Then dfs w here it is that you check what we are doing that you check with the visited. This dfs v that visited v is 1 I am setting it first. Now for all vertices of w adjacent to v, you see whether w is visited or not if not then you call dfs w otherwise, you go to the next adjacent vertex of v and you check again and so on. So, that is the dfs of depth first search and this algorithm is where the unit it prevent the q.

So, with these we like to stop or leave the here today and we assume that we are discuss the different paradigms on sequential algorithms. So, next class will be discussing about that need of parallel algorithms, the 5 types of parallel algorithms and will start designing. So, other differences.

Thank you.