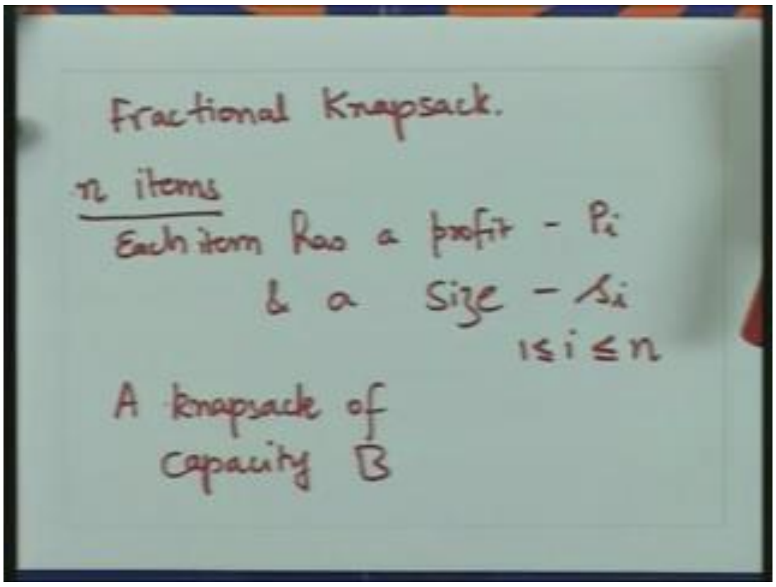
Design and Analysis of Algorithms

Prof. Sunder Vishwanathan
Department of Computer Science Engineering
Indian Institute of Technology, Bombay

Lecture - 12
Greedy Algorithms – III

(Refer Slide Time: 01:03)



Fractional Knapsack.

n items

Each item has a profit – P_i
& a size – s_i
$1 \le i \le n$

A knapsack of capacity B

Next problem, we want to discuss is called Fractional Knapsack. You want to be correct; you can call it the fractional knapsack problem. Well, imagine that you are a bugler and always wanted to start in algorithms lecture this way. So, here goes, so imagine you are a bugler, it is say successful one at it. You set up to bugler house and you have a bag with you or a knapsack. That is why, the word knapsack comes on.

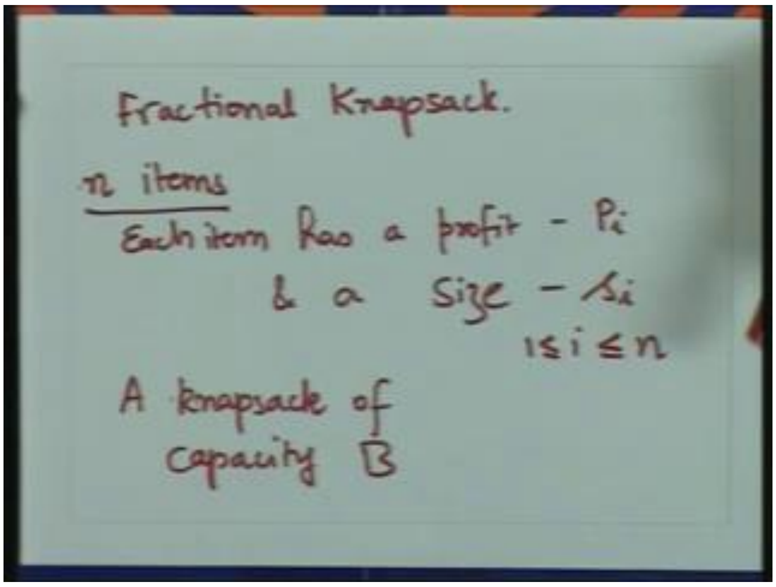
So, you have a sack with you and you enter the house. So, you are a ((Refer Time: 01:45)). So, you enter the house. Then, you find that, you are really hit the jackpot in the sense that houses all kinds of stuff in it. But, you see your sack has limited capacity. You cannot just take everything away, which is, where in the house. So, the crucial question, there you have to answer at this point is, what you stuff into your knapsack? What do you stuff into your back?

Now, each item as a size and there is some profit associated with it. So, if you sell it, there is some cost of profit, that you some money, that you get out of it. Each item has a profit and size. Now, the total amount of item, set you can take back with you is, restricted by the size of your sack. Your objective is to fill this sack with items, such that, the total profit is maximized.

So, you like to somehow decide, which items to put into this sack. So, that the profit is maximized. Well, this is the knapsack problem without the fractional part. Well, if you put take fractions of each item, then it is called the fraction in knapsack. In the sense that, imagine that, you not bugling an ordinary house, you have bugling cake shops. Then, there are all kinds of cakes, fancy cakes tune around and you can take fractions of each cake and select you cannot eat it.

So, you cannot have the cake and eat it too. So, you have to sell this cakes, you can take a fraction of each cake and you like your job is to sort of choose, those which can fit into your bag and make sure not to squish them. And you will again to maximize profit. So, this is the fractional knapsack problem. So, let us get back to our real occasion, which is designing algorithms and let me define this problem formally.

(Refer Slide Time: 04:08)


Fractional Knapsack.

n items

Each item has a profit - P_i

& a Size - s_i

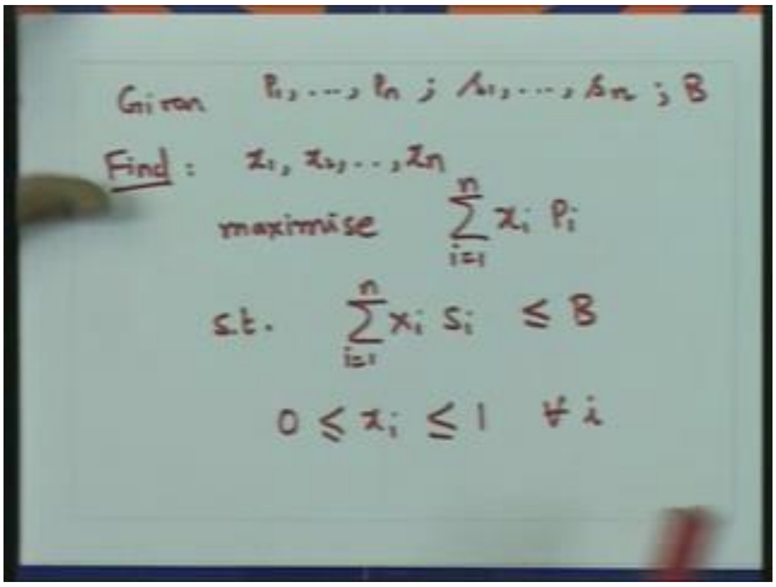
$1 \leq i \leq n$

A knapsack of capacity B

So, there are n items. So, each item has a profit and size, associated with it. So, this is P i and this is S i. Let us say, 1 less than equal to I, less than equal to n. Each item has a profit P i and size associated with it. You have knapsack of capacity B say, this is input. So, this is the input, which is each item has a profit and size and knapsack of capacity B. And the output is, for each item, you have to say, what fractions of this item, you going to take.

So, you are given the each item, there only unit, one unit of an item, which is available to you. You have to take a fraction of this item. So, each item, you have to say, what fraction. So, that the total size is at most B, you cannot take, more than what knapsack, the knapsack and contain and you want to maximize the profit. So, let us again write this down mathematically. Let me recall, you are given.

(Refer Slide Time: 05:56)



Given $p_1, \dots, p_n$; $s_1, \dots, s_n$; B

Find : $x_1, x_2, \dots, x_n$

maximise $\sum_{i=1}^{n} x_i p_i$

s.t. $\sum_{i=1}^{n} x_i s_i \leq B$

$0 \leq x_i \leq 1 \quad \forall i$

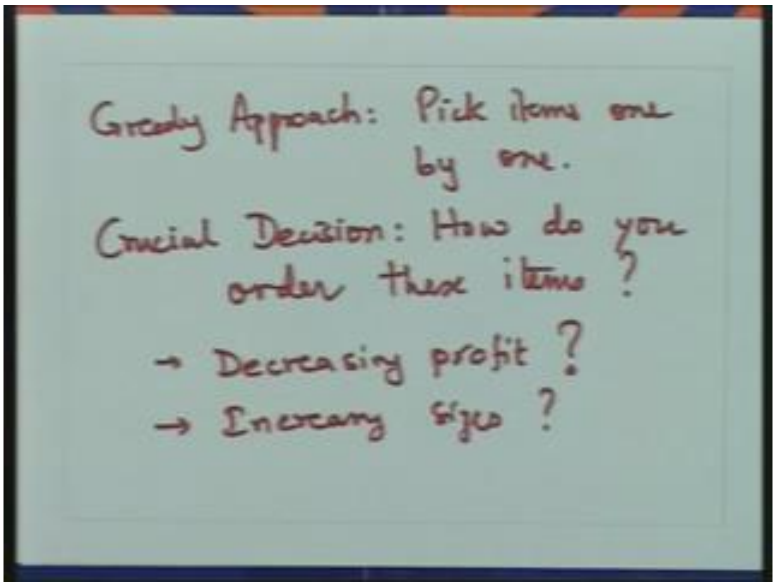
So, you are given P 1 through P n, S 1 through S n, these are profit and sizes and B. So, this is the capacity of the knapsack. What do you want is, so you are going to find x 1, x 2 up to x n. That is, what for action of each item, that you want to pick. So, what should be satisfy, well you want maximize the profit. So, for one unit of item 1, the profit is P 1, for x 1 units, it is x 1 times P 1.

For instance, if x 1 was half, the profit work P could be t 1 by 2. So, you want to maximize sigma x i p i, I going from 1 to n. And the total size of these items; that you pick should be at most B. So, that, sigma x i s i is less than equal to B, i equal to 1 to n. And let me remind you, that x i are fraction. So, 0 less than equal to x i, less than equal to 1 for all i.

So, mathematically, this is the problem you want to solve. The input is P 1 through P n, S 1 through S n and B. Output is x 1, x 2 up to x n. You want to maximize this profit function, which is sigma x i p i. And the x i should satisfy these two constraints, which is, first one is this, that the total size that at most B. And the 2nd is, they must live between 0 and 1. So, this is the fractional knapsack problem. This is the problem; that you like to solve.

So, how does one, go about doing this, where they still within the real muff greedy algorithms. So, I guess, one just has to be greedy and pick as many cakes as one can. So, what you like to do is, pick these items 1 by 1. So, that is what a greedy approach to this will be. So, the output is, roughly looks like a subset. It is a fractional subset not quite subset in the real terms. But, still we would like to do this greedily, in the sense that; we would like to pick elements by 1 by 1.

(Refer Slide Time: 08:48)



Greedy Approach: Pick items one by one.

Crucial Decision: How do you order these items?

→ Decreasing profit?

→ Increasing sizes?

So, we would like to use the greedy approach, which means pick items 1 by 1. So, the crucial question here is, the crucial decision that you want to make is, how do you order items to pick Well, I can come up with all kinds of orderings. You can say, so here is 1, for instance, decreasing profit. So, you could look at items in decreasing profit, according to decreasing profit and pick them, well increasing sizes.

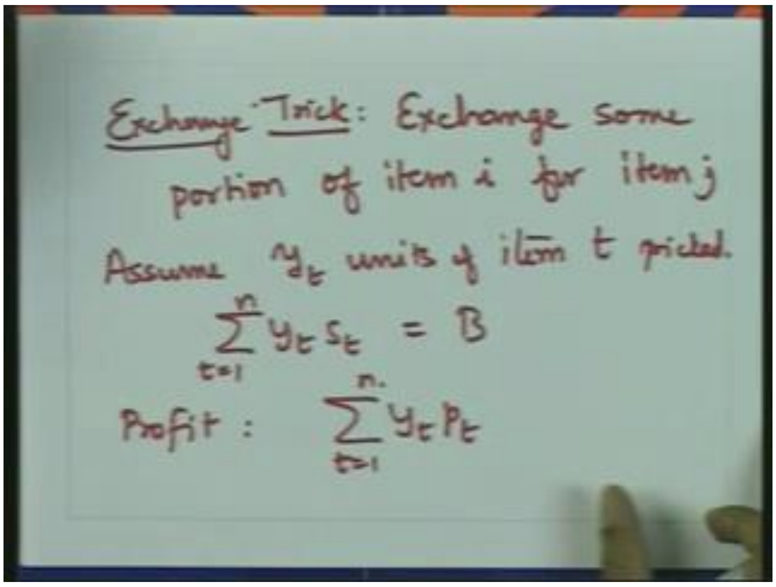
And you could sort of put profit and sizes together into various functions. And let us say, profit by size or profit by size squared, profit squared by size, etcetera, etcetera. And you know, you can ask, which of these work. You can try, all of these, do these work. Well, that is one way of set of approaching this. Try all these, you know various possibilities that go through your hidden, do not go through your hide.

And you know, trying prove that, these possibilities work. Remember that, coming of the possibilities is not too difficult. You just listed out many, coming out with the right kind of order is the crucial, is the sort of key here. And so, how do you do this. One is, when we try the sort of one way is to you know, come up with these things and see, if you can prove one way or the other.

Either come up with examples, so that the strategy works of fails. I mean strategy fails or you try and prove somehow, that this strategy works. What we will like to do is, to use this in defined exchange trick; that we mentioned. And see, we can use act to design, this algorithm. So, that is what we going to do. So, we going to take this exchange trick somehow, you know exchange item.

Basically, put one item in and exchange one item for another and see what happens to the profit. And that, hopefully will guide us, as to how to come up with this order, ordering on a limits. This does not always work, but in cases, where it works, it works. So, let us try this. So, supposing, let me raise, write this term. So, we would like to use the exchange trick here.

(Refer Slide Time: 12:11)



Exchange Trick: Exchange some portion of item i for item j

Assume y_t units of item t picked.

$$\sum_{t=1}^{n} y_t s_t = B$$

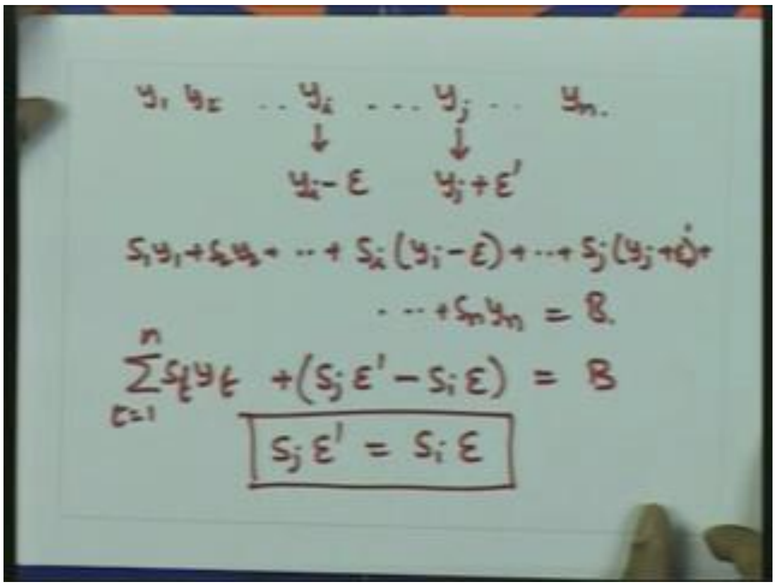
Profit : $\sum_{t=1}^{n} y_t p_t$

So, remember, this is just the way to design the algorithm. This is some kind of trick that you can use and I am not guaranteeing that this will work. Always, it is just hints that I give you. So, try this, what does exchange trick looks like, we would like to exchange, some portion of item i for item j. See, there are two items i and j. I would like to put more of item j and take out some amount of item i.

Then, I will like to see, what happens to the profit. Of course, I have to make sure that, the bag does not bust. So, we can assume that, you know you started out with solution in mind. And you know, you can assume that the bag is full. If it is not full, I can put, you know, I can start filling in arbitrarily some items in the bag is full. So, we take, let us say y 1 of item 1, y 2 of item 2 is so on, y n of item n, such that, the bag is full.

And from now, with this in mind, now I will exchange one item for other for another. So, that the bag still remains full. And then, we will see what happens to the profit. So, that is the sort of calculation that I would like to do. So, will assume y t unit is of item t picked. And so, what is the, we will also assume that sigma y t s t, t going from 1 to n, it is exactly B. So, the bag is full. So, what is the profit, the profit is nothing, but sigma y t p t, t going from 1 to n, this is the profit. Now, what would I like to do with this is, remove some of item, some amount of item or high an increase some amount of item j.

(Refer Slide Time: 15:06)



$$y_1\ y_2\ ..\ y_i\ ...\ y_j\ ..\ y_n.$$

$$\begin{tikzcd} y_i \arrow[d] & y_j \arrow[d] \\ y_i - \varepsilon & y_j + \varepsilon' \end{tikzcd}$$

$$s_1y_1 + s_2y_2 + \cdots + s_i(y_i - \varepsilon) + \cdots + s_j(y_j + \varepsilon') + \cdots + s_ny_n = B.$$

$$\sum_{t=1}^{n} s_t y_t + (s_j\varepsilon' - s_i\varepsilon) = B$$

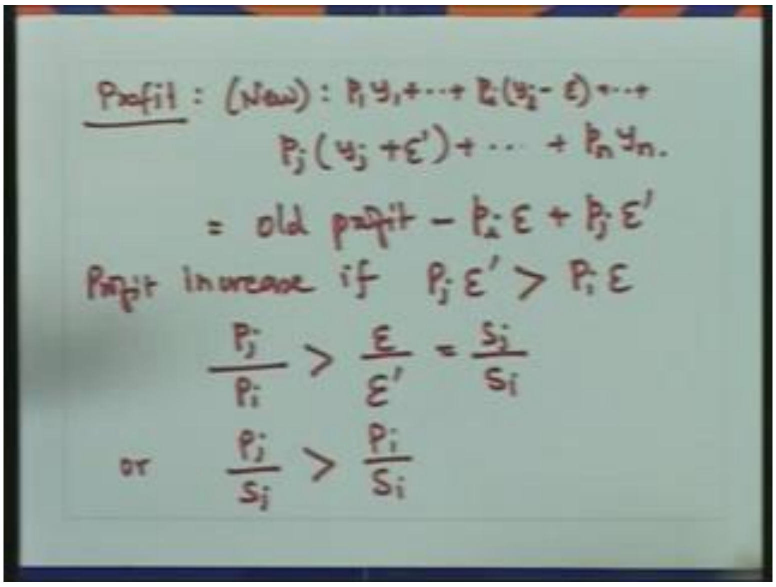
$$\boxed{s_j\varepsilon' = s_i\varepsilon}$$

So, in my new thing, initially I had y 1, y 2, somewhere had y i, somewhere had y j, y n. This is what I have. Now, what I would like to do is, make this, let us say y i minus Epsilon, for some small Epsilon. And this, I would like to increase by some amount. Let us say y j plus Epsilon prime. This is what I would like to do, which is I have replaced some portion of the ith element with some of the jth element. I would like to maintain the bag to be full, what this transactions to be such that the bag is still full.

Then, I would like to see what happens to the profit. So, what it mean that the bag is full, it means, S 1, y 1 plus S 2, y 2 and so on, plus S i and y i minus Epsilon plus 1 S j into y j plus Epsilon plus S n y n equal to B. So, this is sigma S i y i sorry, S t, let us say y t, t going from 1 to n, plus S j times, this is Epsilon prime. Epsilon prime minus S i times Epsilon equal to B.

Well, this fellow I know already B, because, initially I started out with the bag being full. So, we have that S j Epsilon prime equals S i Epsilon. If I want the bag to be full, then I guide that Epsilon prime and Epsilon I can choose in the ratio of S j to S i.

(Refer Slide Time: 17:26)



Profit : (New) : $P_1 y_1 + \cdots + P_i(y_i - \epsilon) + \cdots +$
$P_j(y_j + \epsilon') + \cdots + P_n y_n$.
$=$ old profit $- P_i \epsilon + P_j \epsilon'$
Profit increase if $P_j \epsilon' > P_i \epsilon$
$$\frac{P_j}{P_i} > \frac{\epsilon}{\epsilon'} = \frac{S_j}{S_i}$$
or $$\frac{P_j}{S_j} > \frac{P_i}{S_i}$$

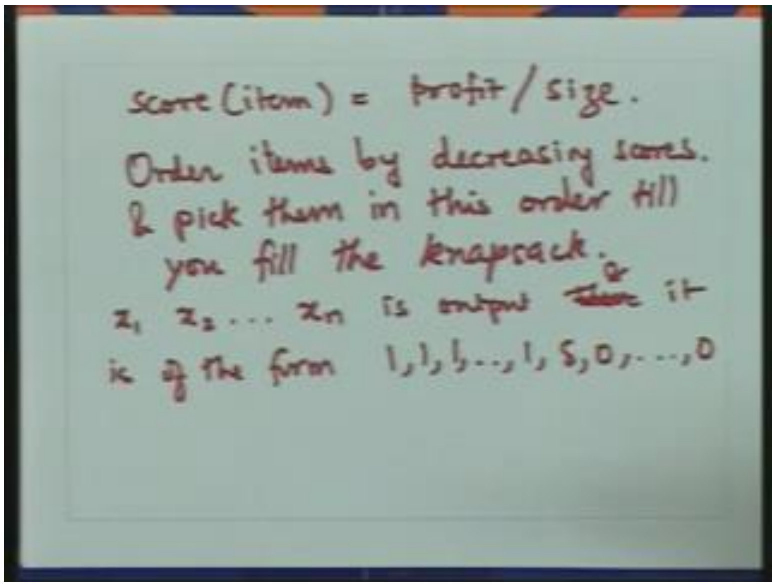
So, let us see, what happens to the profit. So, what happens to the profit? So, what is the new profit? The old profit we know. The new profit is, new is P 1, y 1 and so on, plus p i into y i minus Epsilon. So, on plus p j into y j plus Epsilon prime and we go on p n times y n. So, this is nothing, but the old profit which is P 1, y 1 and so on, p i y i etcetera, p j y j plus p n y n plus minus p i time Epsilon plus p j times Epsilon prime.

So, the new profit is nothing, but the old profit minus this plus, that good. So, now the profit, there is a profit increase this is. So, this is what we want. When, is there profit increase, well this fellow is greater than 0, which means p j times Epsilon prime is greater than p i times Epsilon, which means p j by p i is greater than Epsilon by Epsilon prime.

So, there is a profit increase, this is 2. Let me just full back and old slide ((Refer Time: 15:06)) on this is s j times Epsilon prime is s i times Epsilon which means Epsilon by Epsilon prime is s j by s i. Now, so let us just put this bag into that other inequality. We are going to put this equality into other inequality. So, this Epsilon by Epsilon prime is nothing, but s j by s i.

So, there is a profit increase if p j by p i is greater than s j by s i or p j by s j is greater than pi by s i. There is an increase in profit if p j by s j is greater than p i by s i. This tells you, what to favor. I remember this happen, if I replace part of item i with item j. I should favor item j if p j by s j is greater than p i by s i. That is what this calculation tells and that is what my algorithms will give.

(Refer Slide Time: 20:58)



score(item) = profit / size.

Order items by decreasing scores.
& pick them in this order till
you fill the knapsack.

$x_1 \; x_2 \ldots x_n$ is output ~~then~~ & it
is of the form 1, 1, 1, ..., 1, S, 0, ..., 0

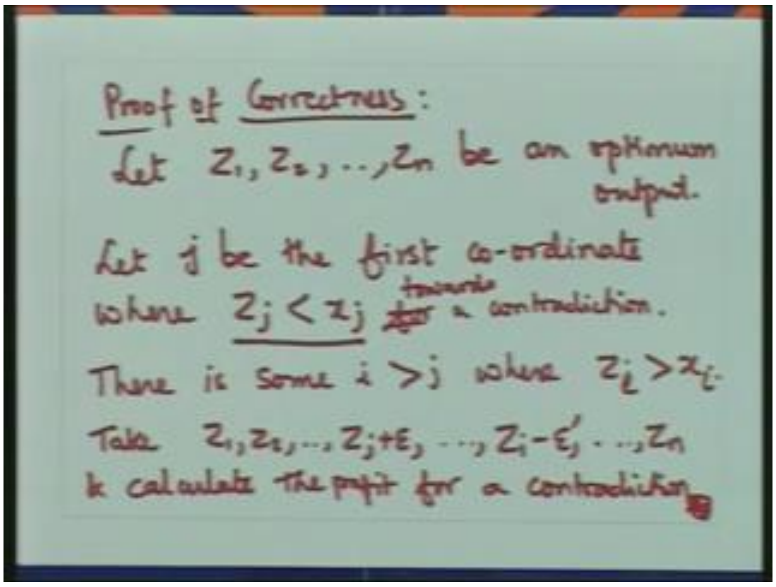
So, you can see, how just using the exchange trick. We have actually device an algorithms and we will, in fact, also proof that this algorithm is optimum. So, these score of an item is nothing, but profit divided by size. And the algorithm is simple, you say, the algorithm says order items by let us say decreasing scores. And when pick these items in this order and pick them in this order, till you fill the knapsack. This is the algorithm in one sentence.

So, how will they output look like, well you pick the first item, the one with top score, you pick the item, you pick the next item and so on. And at some point, you know you picked these items and the next item; there may not be enough space in the knapsack. So, you do not pick the entire item, you pick a fraction. So, the first few items, the first, if I pick k items, k minus 1 item will be, I will pick the entire item unit. And the kth item will be a fraction.

So, in my solution, there will be one fraction, which is a last item and the rest of them will be units. That is what the outputs looks like. So, the algorithm, if I look at, let us say x 1, x 2, so on, this is the output. And let us say 1 is the item with maximum score. So, this is items by decreasing scores. This is the maximum score and so on. This is the least score.

Then and this is the output is output. Then, it is of the form, this sequence is and let us say, it is of the form 1, 1, 1 up to some level. And then some let us say delta, which is between 0 and 1, and then 0, 0. This is how the output looks likes. We need to now proof that, this in the works. This algorithm, the output that this algorithm gives you is optimal. So, let us prove that this is optimum.

(Refer Slide Time: 23:51)



Proof of Correctness:

Let $z_1, z_2, \ldots, z_n$ be an optimum output.

Let j be the first co-ordinate where $\underline{z_j < x_j}$ ~~for~~ $\overset{\text{towards}}{}$ a contradiction.

There is some $i > j$ where $z_i > x_i$.

Take $z_1, z_2, \ldots, z_j + \epsilon, \ldots, z_i - \epsilon', \ldots, z_n$ & calculate the profit for a contradiction.

So, we would like to compare this with any optimum and shows that, this is optimum. This is the, next way we would like to do it and we would like to use an exchange trick again. But, now to prove that it is indeed correct. And we will use it the same way, we use it earlier. So, please set of review that material, if you forgotten. So, the proof goes this way.

So, supposing, so let z 1, z 2, z n be an optimum output. Let me recall that, our outputs looks like this, ((Refer Time: 20:58)) it is x 1, x 2, x n. It looks like 1, till something. These are in decreasing order of scores. Then, you have a delta and then 0's. This is how our outputs looks like good. So, let z 1, z 2, z n be an optimum output. Again, the items are, the order of the item is at the same.

In both, z 1, the item 1 is an item with the top score. This is the way the optimum picks, if it is of course, 1, 1, 1, and then, delta and 0. Then, you have done, because this exactly equals z exactly equals x. Now, let it is possible that z 1, equal to x 1, z 2, equal to x 2. But, if they are different, they will be first index, where they differ. There will be a first index, where they differ.

So, what you are going to do is, we picked more of these item in the first index exit ith index they differ. Then, here we thick more of I and he picked less of I. Because, we picked one unit of I and he is picked, let say, less than one unit of I. Then, we would like to see what happens. If we increase, we take the optimum increase the ith and we decrease something some other, where which as a lower score, we can do this.

And using the previous sort of calculation that we did during the exchange trick, we will see that the profit goes out. This is the argument. So, let us just formalize. So, let j be the first coordinate, where z j is less than x j. This is also the first coordinate, where they differ. Well, I will let you observed or proof that, this first coordinate, where they differ, z j will be less than x j, it will not be greater than x j.

That is the way, we have picked exist. They are once still some stage and then, there is a delta total sum is B. If I increase any one of them, it is cannot be, the once I cannot increase obviously. And if all the initial things are 1, I cannot increase this delta. Because, then it will be greater than B. So, the first coordinate, where they differ z j will be less than x j.

So, then there is some i, which is greater than j, where z j is greater than x j, z i is greater than x i, some x i, where this is true. Because, the total of both of them is B. Now, what I do is, I increase the profit. So, one can increase the profit here by doing this. So, take z 1, z 2, z j plus Epsilon and so on, z i minus Epsilon and so on z n. So, look at this Epsilon prime, do the same calculation that you did. And you can see, that the profit, remember as an end of it, we came up with the new profit, which is greater than the old profit.

Of course, we need to choose Epsilon and Epsilon prime properly. And I like to look at, I like to do this. So, go back to this old calculation that we did. Figure out, what Epsilon the Epsilon prime should be. I again have enough items to fill enough type size of B. With the profit as strictly increased, which means at the original z 1, z 2, up to z n was not optimum. That is the contradiction.

So, this is, for a contradiction and calculate the profit for a contradiction. I am said towards, that is the end of the proof. I let you sort of frame this better and write it properly. That is one small thing about this proof which I should say, supposing, so let us say that, these items are same score. They all had different scores and then, there is no problem in this proof, it was perfectly.

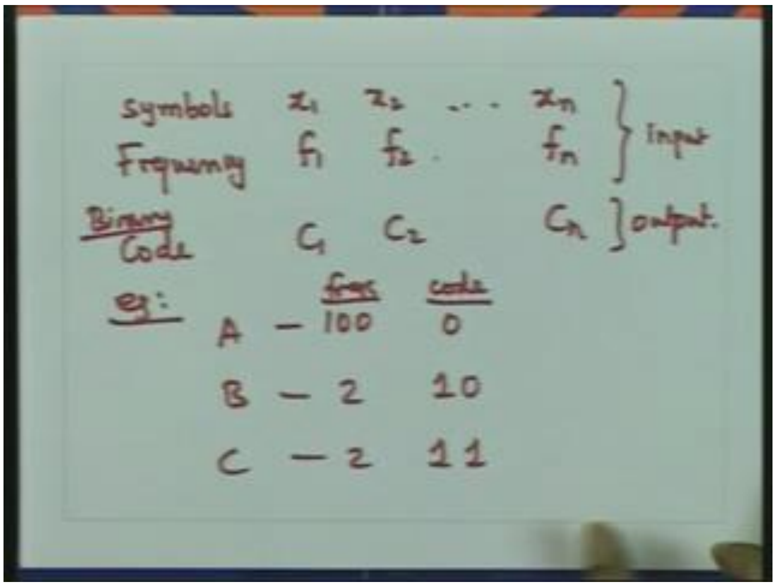
If the scores are the same, then you need to worry about it. You need to modify the proof slightly. Basically, I could up chose in you know an item with, among items in the same score, I can sort of chose any one of them write. And I may differ, from the optimum in this respect, but it is only in this respect. That part of the proof, I will let you sort of film, but module that, this finishes the proof of correctness of the algorithm to fill up knapsack.

So, again, let me emphasis this, that the right order that we want is to pick to look at the profit by size. This is also; you could up come up with this intuitaly also. You know, your intuition could have told you, in fact, even I mentioned initially, that you look at profit by size, as I measure, which we should follow. But to sort of the way, we got it was just by trying out this exchange idea.

And just exchange two things and we saw what happened and profit by size, came out naturally from this from this exercise. So, that is just see, that is the model that I want to commit you. That often, when you are face the problem this kind. Just do the exchange trick. Run over this exchange tricks; see, what is the quantity that comes out. And that quantity, hopefully, will let you dictate, what your algorithm should be.

The next problem that we look at comes from the area of information transmission and it is related to codes. So, we could like to design codes, which have some properties. That is the next problem.

(Refer Slide Time: 32:50)



symbols	x_1	x_2	...	x_n	} input
Frequency	f_1	f_2	.	f_n	
Binary Code	C_1	C_2		C_n	] output.

eg:

		freq	code
A	–	100	0
B	–	2	10
C	–	2	11

So, the problem is like this. So, you have symbols x 1, x 2 and so on up to x n. So, you have n symbols, each of them has of frequency f 1, f 2, f n. So, these are symbols that as sent on a channel from let us say place A to place B. These are symbols, which are same from one place to another and this is the frequency at which they are send. And what we would like, so this is the input. So, what we would like is codes c 1, c 2. We would like to code them it is using c 1, c 2 up to c n.

These are, let us say, binary codes. Each of these is a binary string and we would like this binary strings associated with each of these symbols. So, for instance, if I send c 1, c 1, c 2; it means, you have received, you send symbols x 1, x 2; the word x 1, x 1, x 2. So, it sends these symbols, I just send this codes, which a sort of decoded at the other end. And what you send is not symbol at a time, but huge word at a time.

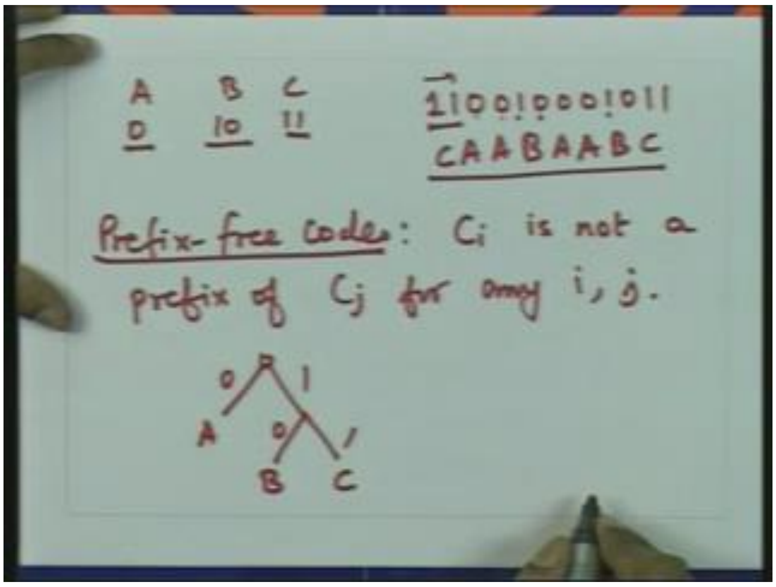
So, you have this huge sort of string, which keeps sending from the left. Another right, you sort of full of those strings and you decode it. So, this code is the output. So, if it not for the frequency, there is nothing much to this. I have n symbols, I just write down; you know and sort strings as scored. Then, I send them out the frequencies that make things are being difficult. So, let us take an example.

So, for instance, I have symbols, whereas, in example, I have let us say A, B and C; free symbols. For frequencies are, 100, 2, which means, it is send 100 times 2nd; B send 2 times the 2nd; C 2 times the 2nd. These are the frequencies that which average frequencies that which a send through. Now, we would like code, these are the frequencies. We would like a code, well here is a good code, a is 0, this is 1 0 and this is 1 1.

So, let us look at this code for A, B and C. Well, I use stands to resend that I would like to use small number of bits for A because, it sends many times. So, I would like to send this smaller number of bits across. So, this has 1 bit, then I have 2 bits for B and 2 bits for C. So, then, it is sort of, I use lesser number of bits, which I transmitted across from from left to right.

Now, this code has another property. So, if I look at the strings of 0's and 1's going across, decoding is easy. The reason is this. As soon, supposing the first symbol is 0, I note it is an A for sure. If I hit a 1, if the first symbol is 1, I just look at the next symbol, it is either 0 or 1,depending on whether, it is 0 or 1, I know it is B or C . So, let us do one example here.

(Refer Slide Time: 36:45)


A B C
0 10 11
11001000101
CAABAABC
Prefix-free code: C_i is not a prefix of C_j for any i, j.
0 1
A 0 1
B C

So, I have A, B and C, 0, 1 0 and 1 1. Let us take a binary string, let us say 11001000101. So, how do I decoded. So, I am getting the symbols from left to right. I get 1, I check, I know it has to be either an A or B or C. So, this gives me C. I get this 0, this has to be an A. This has to be an A, 1 0 means B, 0 is an A, 0 is an A, 1 0 is a B. 1, well there has to be something else, here let us say 11 and that give you C.

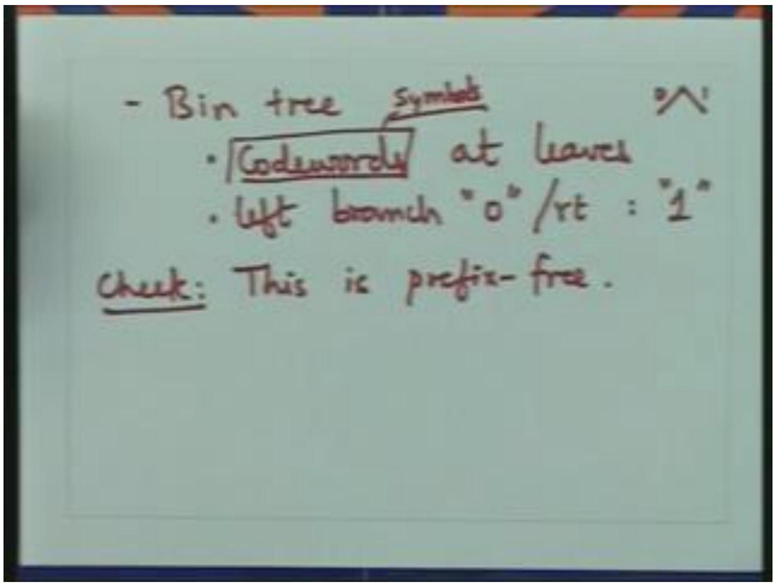
I can just look at this symbol as they arrive the bits as they arrive and decode. Decoding is very easy. This code, which have this actually has a property, which I will tell you, which is that. None of these, is a prefix of another, 0 is not a prefix of any of these and these two are suddenly not prefix of each other. So, such codes are called prefix pre codes.

So, if I have code C 1, C 2, up to C n, C i is not a prefix of C j for any i j, then it is called a prefix pre code. And these, you can see are easy to decode. I just keep looking at the symbols and as soon as I get a code word, I am done. I know, what about is. So, I write the word, I remove these and I keep scanning. So, we would like to construct prefix pre codes.

So, that is an objective, given these symbols, I would like to construct prefix pre codes and given the frequencies. But, each frequency, there will be a, with each alphabet, there is a frequency. I would like prefix pre codes, which minimize the average in your length of the message with send, which means, the length of a message, I will think of, the length frequency times, the length of the code word.

It has the code has some length l. So, the frequency times length is a length is a average length. And the some over all elements I would like to minimize. So, before I state this formally, let us observe that, prefix pre codes have a close connection to binary trees. So, for instance, this code A, B, C, I can represented this way, 0 1, 0 1, A, B, C. So, basically I want to show that, given any prefix pre codes, code, there is a binary tree and given any binary tree like this, I can get a prefix pre codes.

(Refer Slide Time: 40:05)



- Bin tree Symbols
  - Codewords at leaves
  - left branch "0" / rt : "1"

Check: This is prefix-free.

So, what is the secret of this transformation? Well, take any binary tree; the crucial things are code words at leaves, the leaves, where you look at the code word. Left branch is 0 and the right branch is 1. So, when I take the left branch, I think of it has 0, right branch I think of it has 1. And when you reach a leaf, you know, you have traverse a binary string. Now, that will be the code word, associated with that with that symbol.

So, for by go left, left, right, left and there is a there is a code word there, will be 0 0, 1 0. So, given a binary tree, with these code word, at leaves, with these words at leaves. Then, I can assign a code word to each of these symbols. So, maybe, I should say symbols, given a binary tree with symbols at leaves, I can assign a code word to each symbol, using this 0 and 1 business.

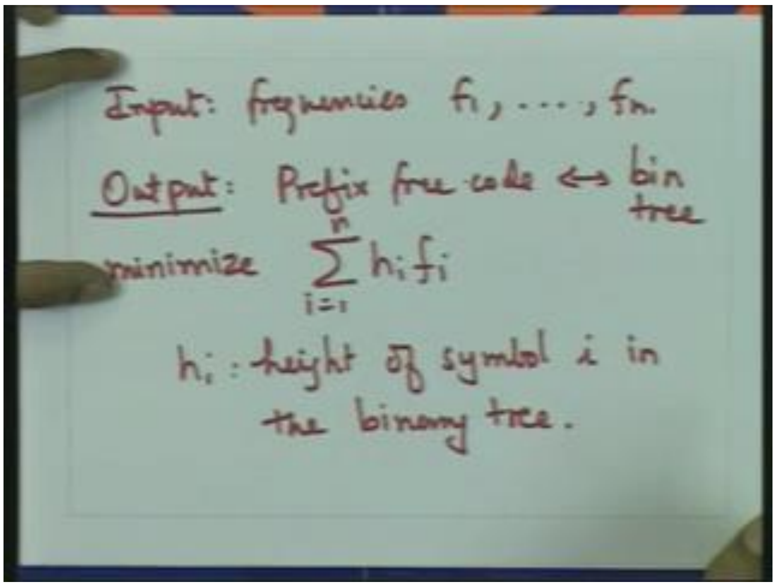
You can check that, that this is prefix free. So, why this so, because take any node in the tree. So, this has some binary string, associated with it, which is you followed the path from the root to leaf. And if you go right, you have a 1, if you go left, you have a 0. So, you get, there is a binary string associate. Now, what are the prefix of this binary string? The prefix are exactly those string, which you get at nodes, which are in the path from the root to this node.

If I only take leaf of a binary tree, no tree, no leaf is a prefix of another leaf, no leaf sets on a path from another leaf to a root. Let us say just not on. So, the codes that you get this way of prefix free and given any prefix pre code, you can construct the binary tree.

Just the simple fashion you just go 01, 01 at each stage and you follow your follow this 0 and pattern.

And when you hit a pattern, which you want, just remove everything below it and that will be your leaf. I am the word associated there with that will be the code word. Will the symbol, which is associated with that code word. So, these are prefix pre codes, what we want is, you want to construct prefix pre code. Before, this let me just take a new set of paper and write this entire problem properly.

(Refer Slide Time: 43:34)



Input: frequencies $f_1, \ldots, f_n$.

Output: Prefix free code $\leftrightarrow$ bin tree

minimize $\sum_{i=1}^{n} h_i f_i$

h_i : height of symbol i in the binary tree.

So, the input is frequencies f 1 and so on up to f n. What is the output that we desire? We desire prefix free codes or in other word a binary tree. And we would like to minimize the following function sigma h i f i, I equal to 1 to n. The frequency is come from there. Well, things with large frequencies must be very, must have sort codes, which means they should be close to the root from the tree.

So, h i in fact, is the height of symbol i in the binary tree. So, the input of frequencies f 1 through f 1, f n, I would like to output a binary tree. And with each leaf, I associate one of these works either1, 2, 3 up to n. One of these symbols and I would like to minimize h i f i with the some h i f i, where h i is the height of the symbol I in the binary tree. F i is the frequency of symbol I. So, this is what I want.

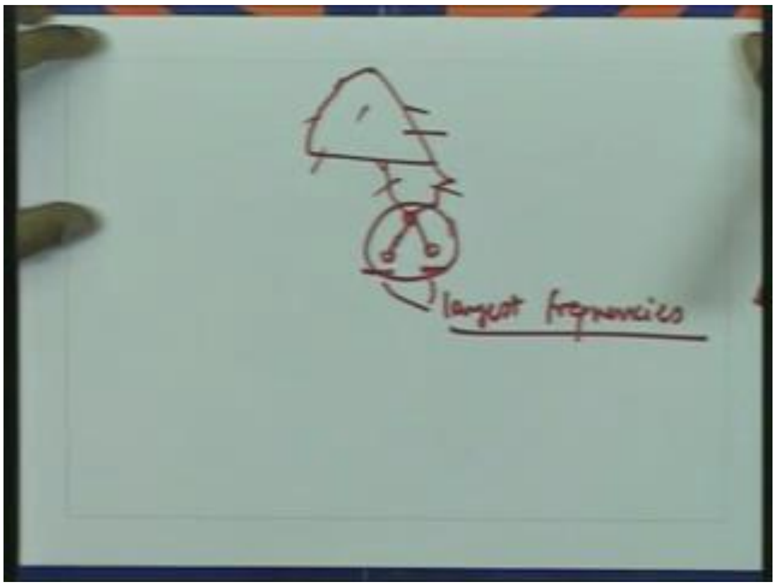
Now, how do I go about doing this? So, given a binary tree, supposing that, I give you a binary tree with n leafs. Now, can you assign these words to this leaves. Well, or to minimize this function, when the answer is, it is easy. So, given a given a tree, since I want to minimize the h i f i, I put items with the largest frequency as close to the root. So, basically, I take these leafs, I sort them in decreasing order of height.

Saying increasing order of height, I take frequency, I sort them into decreasing order of frequencies and I just tied them up. So, if the tree is given, then this is easy. The heaviest item of the most frequency must set close to the root and those with whose frequencies of the least will sit for just away from the root in this tree. The question is, what the tree should look like; there are many binary trees with n leaf.

It is a good exercise to see, how many. If there are large number of binaries trees with n leafs, which of these threes we pick. Once, you pick a tree we are done, we know, how to sort of fit these symbols into the leafs. So, how do we pick these trees? Well, do you like to do something some kind of exchange trick here. Even here, what could the exchange pick really mean, you know, you want to come up with the binary tree. And you have these items, which set of these symbols, which are sitting.

What does mean to pick? What is the exchange trick mean? So, what we would like to do is, construct the correct tree. That is, what we want to true, you want to do. And so, what does it mean to do an exchange trick. So, supposing you have some tree, you know that the bottom of the tree, which means node with the longest height. You have the two items with the largest frequency, will sit there. So, this much, we know.

(Refer Slide Time: 48:07)



So, we know, we really know some part in the tree, which is, this is a binary tree. Now, the bottom most points, somewhere here it is a bottom most points. I have, once with largest frequencies. The two items with the largest frequencies will sit at the bottom on the tree. The other leafs are somewhere there. This much, I know. So, I know part of the tree.

What about the rest of the tree. Can I sort of somehow compute the tree part by part, well, so that is one thing. What could an exchange trick look like here? What do I exchange items. But, if I exchange items, remember the tree remains the same. So, the trick, the exchange trick that you would like to try is exchange sub trees. So, given a tree, I pick two sub trees and exchange them and see what happens. So, this is the calculation we would like to do. And once we do this calculation, we would like to see, what really happens to the average height. And from this hopefully, will get a clue as to how to construct in the tree.