### Algorithms\_week3\_lecture2\_1-20240914

说话人1 00:45  
I guess it will start. Now, as they will first finish the last example, what do I conquer from the last picture? And then we'll start a new section. Now, the last example is called maximum suffering problem. And then the second one is about stock buying seven. So you can see that the in the last line, there are some numbers. And these numbers. The first line is the number of days. Then the second line is the price that if you buy yourself on day four, you need to pay 105. And supplying price and selling price is the same.

Now the question is, if only buy months and sell once, then on which they will buy on which they will sell, suppose you can only buy one units of stock. You have enough money, but you can only buy one unit stock, and you will buy ones and sell ones. You want to earn the largest possible profit. So what to do with this problem? So any idea? Quick thinking. Now, the basically, you try to find the lowest price and highest price and then do the difference, right? So the very natural method will be you first choose the highest price. Then before that, you choose the one equal to the lowest level. Okay? Then you buy at that moment and sell at this highest price. There's a symmetric way between the you find a little bit fun, and they buy from that day, and then you sell it in the future. So which day on yourself is very easy? It's just the highest point you ever reach right in the future days.

These two methods, they are symmetric, but unfortunately, none of them is optimal. So you can see this example in this example. Where is the lowest .? Two or four. There are four. Actually, there's cinema here, which prices was prices 60 k so lowest price is four. If you buy here, you have no chance to sell it back. That's not good. And then if you say I want to sell at a higher point, they have to buy on day zero, but you only get $1, right? I do this. What's the best solution here? Two and three, right? So by doing this, you will earn $3, right? Okay. Now you need to find a way, an algorithm which can give you this conversation.

Okay? Now, one easy way is, again, enumeration, right? Because you only have two decisions to make right on which the violence is that when they just try all the combinations, the best of them must be your solution, right? Ok so by doing this, you need n squared time. Right? Ok it's correct. That is too. So we do better. Now you need to do it better. We probably need to transform the problem in some way, so that it's easier to describe the problem, because currently, the problem looks like that. You need to decide 2 points, right? And then buy at some point and sell at some point and focus on difference. This one is not very easy to encode into aaa nice mathematical problem, although you can also still do that, but a it's not so easy.

So now, let's change the problems in the far away. We no longer focus on the stock price. Instead, we focus on the price difference.

See that on this picture, we put in a new line, this new line, we've got a change of price. For example, the new line, the first number here is what? 113 minus on that, right? The difference is 13. Then similarly, 110 - 113, the difference is - 2, right? So we just recall all the differences or adjacent days, right? And put it in a new life. So now, what's the problem? The problem now becomes, so originally want to find it buying on the same part. Right? Largest problem. Now, you can talk I want to find a side ., any point such as the differences are they add up to a little bit back? Right? Is that right? Now it becomes a suffering. I want to find a separate of the third line such that elements in the suffering. They add up to a value as big as possible.

This is so called the continuous suffering with them. Or because continuous subject rain, you can consider it a kind of a concept like substrate. You mentioned last time. It should be well, without any break in the middle rank. And then we try to find a continuous separate with the maximum sum and call it the maximum suburb.

Now, to do this, we can adopt a divide and conquer solution. What is that? Now remember, last time we do divide conquer, it's about this problem. Which one? The name of the problem last time it is the case, not just fine. You have to swap the two words, right? The case minus number. Or in other words, we call it selection problem, right? This is the first example of development we do last time.

Now today this is also about conquer, but now we are trying to find a maximum separate. We want to find the two indexes, right? Also, sorry, we are looking for all right, looking for a maximum sub array within this range. The lower and higher are two indexes. This is the boundary you cannot go beyond. I'm trying to find the maximum array in this range. It could be here. I have to be somewhere inside you. Okay?

Now, when we divide the problem, we very naturally is to divide in the middle. So we divide the whole array into 2 pieces. 1 piece is from the lowest point to the middle point. Middle is just half, right? And then the other one is from middle point to the highest, yes, ok two pieces. And then we will see three possible situation for where this maximum salary is. Okay? So which three? Now the first situation is, the maximum sub array is in the left piece, is here, somewhere here. Here's one. It would also be somewhere here on the right case. And then it also cross this middle point, right? So the best separate and maximum separate to cross the middle line. Altogether, you have three possibilities.

Now, for the first two is quite easy, right? Because for the first two, you just do recursion, right? You call the left piece. Please tell me what is your largest summary, right? And that also would be the answer, right? Okay. If it's on the right, it's also straightforward because you just dropped it for the right side problem. And you guess. But if the solution that the final answer crosses the middle ., then recursion cannot help you, right? Because they can only carry individuals separate on either side that it crosses. But in the process, you can see some good feature of this separate crossing this .. What's the feature? Let's see. The future. So the the good feature, sorry. Yeah, so too much.

So basically, I already explained the previous pages and don't worry much. So for the third case, right? Crossing the middle point, then actually, what you need to worry about is just starting from I until middle point. I just need to find this piece and also that piece, right? So you don't need to worry about, it's somewhere in the middle. It's somewhere in this middle, right? So you don't need to worry about it somewhere here. Because if it the final answer process, this middle point, then the left end . until middle, you have to take it, right? So the final answer must be on the format that touches the . middle. It has to touch middle.

Now, if you touch the middle, then what I do, when you are given this kind of range, you have to find the best operate in this range. Then the only thing you need to do is you try from the first point to middle. What's the sum? From second point to middle was the sum, right? Third point to middle was the sum. I mentioned, all these sums and take the light. That must be your best choice of the separate crossing the . on the left part.

So this is middle, right? This is, this is 200, this is 90. So then I will be probably right. This will make for some very big. Right? And no matter what choice you make on the right side, it will not affect your choice on the left, because left on the right now are independent, right? Your choices on the left piece and the right piece, they do nothing, do not affect each other. Right? Ok so they just make independent decision for two sides, and then combine the two sides together. So what's the right time here? The only thing you need to check is a study for midpoint. You can do this, right? One number is a sum starting here, right? Then you add another number. It becomes sum with two numbers. Then you add another number. It becomes sum with three numbers. Altogether, you just need a linear time, everything to check all the sounds and find the best sound, right? And they answer them. That means when you do recursion, right? When you do this recursion, the time spent finding the middle piece is linear time. It is proportional to the size of the array.

That's easy, right? What's the running time is here? Now, with the knowledge from last time, you may know this tn is the time to find the best array or maximum sub array for an array of size n then this part is what? Two times tn over two. Because left half and the right half, each of them has size n over two. You have to solve each other, right? So that part, those two matter, this two time, these two has an easier, meaning compared to the two. We talked about last time for selection from. Right? Remember, the last time we have two times t two anniversary, right? But those two, two n over three has different meaning. But now here, tno two, they are the same meaning. Okay? It's just solving the two sub problems.

Now. What's this? Yeah. It's the time to find the best one crossing the middle ground. Ok your recursion just looks like this. Tn equals two times tn over two plus on then this one is exactly the recursion you will see for the very classic merge sort error. If you have done it, that's good. You didn't learn. It's still okay. Anyway, it's sorting, right? So it's aa merge sort with the same recursion. And then the running time for this, pn will be on n log n it's n log n time to find the maximum separate. This is the first metaphor. Sorry, not this, right? The second measure to solve the maximum separate problem, the running time is better than when squared.

Now, this method was discovered by michael seans when he's teaching in cfa okay?

Now there's a very funny story, although in the be a bit exactly a it. Basically, I have not believe that 1 minute. So. But anyway, so they want student in the class taking the seminar. And then this j today came up with another solution, which is linear time, not non now, here I said in 1 minute. So I would not believe in that. So must be shortened a lot ok then let's see how to improve that. What improvements? We will still use this recursive recursion idea, but we will not use divide and conquer. Instead, we resort to dynamic programming. We learned last time ok so what is that?

So then what it says, now I will look at the numbers from the first number to the nth number, sorry, a case number. So one to table I one to I on j that is in jail. One today, here, one to j right? Now. Suppose from one to ji already know the best answer. What's the maximum separating from energy? Now, is there any way to use this solution to help you solve the problem when you have one number? Okay, so can you do it? I this time alone is the, when I use that, how do you service?

So you have one to j here, 12, j and you have aj plus one. With the answer from one to j can you have an answer for one to j and two j plus one one more number? It would not be that easy, because the solution in this part may not connect with j plus one. Right? Prepare some remedy here, right? It does not touch the mj and then if you include, it does not mean it does not form a summary, right? It's somehow broken. That's not good. Therefore, in jane a canadian solution, he said ok then why not I keep two songs? One is the best summary in this range. The second one is the best separate up to the last element, which means from one to ji also keep the best sum up to the j seven.

So you have many search. All right, many things. All right. I will be the best one. The best is the one to the largest sum. Among all these touching the boundary. I also remember that. I also recall that. Okay. Now I have two guys to keep for every sub problem. One value is the best summary from one to j the other value is the best summary touching energy. Okay? Now, with this, it seems a bit easier to update the value when j comes j plus one. How do we do that? Now I want to find out what's the best sum if I touch j plus one? How are you going to do that? That's something touching data. Can you use the results for the earliest of the payment? Can use that result? For example, I tell you the best array touching j is this one. I tell you the best operator engage this one. Right? There was the success separate dj plus one, still this one. But one more evidence, right? You have to include j plus something, right? Ok so that answer is, it's quite easy to prove, right?

That's where it should be this one together with this editor, at least together. This will be your solution or the answer touching paper, the best already had been as well. Why? It was easy, this larger sum, any of these sums are smaller than this one, right? Or as not as good as this one. So taking this does not, will never hurt you. Right? Now you have the best separate touching j plus one. And then next one, that was the best summary. For the whole range. It costs one. It's negative possibility. You should judge whether the j plus one is negative or positive. So that's why, for example, we should divide our selection into three possibilities, not two. Because if they do not need to worry about this, there are only two possibilities. One possibility is what one possibility is. The earlier answer is computed for this range. So there is the best already who does not touch j plus one? Some idea, right? This is the best operator. This is still one candidate of your new solution, right? This is still one candidate.

Now, what's the second candidate? The second candidate is something that will touch j plus one, something that touched us on this one. We want to do it, right? Okay? But then you think the only two choices. One is, you do not have shape as well, right? The other one is you touch it as well. Remember, just as you just said, touching j plus one may be good, but maybe not good. When j plus one is positive, you should touch j plus one. But if j plus one is negative, you will not take this edge. Are, you will not take this one? No, I will not take this one. Instead. I'll take this section. The one we got it. It depends on whether j plus one is positive or negative, then the one touching j plus one. Somehow you need to differentiate into two cases, right? Either you really touch it as well, or I do not take touch j plus one, I just touch it, right? Among the three terms, you take the best. And that would be the answer for 12, j plus.

Now, so this is exactly what I wrote here. You need to discuss about three possible cases. Now, even three, it is not a big deal, because the computation is this several steps, right? Fixed number of steps. So whenever you include the new number into your all right. You only spend, for example, three more steps to handle the sum. Overall, it will be proportional to the size of a rate. The total number of steps you do you need to compute, right? Is proportional to the size of a rate, which is on right? Or in linear time, l this is the so called improvement in 1 minute, but there are some maybe the first stage they do not formulate the problem in this way.

There is another way because in the session I gave in hong kong, some of the students told me another solution, but that solution actually correspond to the model without transformation. It. Because once the model with only stock prices, same rationale, meaning that the dynamic programming will look at the current price, so the current stock price and keep the solution.

If you sell at the current ., what's the best thing to get? Right? Ok so then it's still kind of a panic program approach, ok so I think that for divide and conquer this stuff here, any immediate questions? Ok so if no, then I would like to remind you something. Throughout the course, you will always see some of the earlier approaches that will appear in future topics. And those will be the most likely bonus points. For you. I remember. Now I will do it today. All right? In the future, I also do it. Now, if you can recall what you learn before, then you have higher chance to get all this. All right? Basically trying to establish a link between the knowledge learned before and the knowledge of learning instead. Okay, so this is one thing. And then also in this 3 hour session, please think carefully or think actively, because many of the examples we talked about in the lecture, they are accessible to everyone.

So they are not super difficult things. All right. So trust yourself. Now, let me open the. Today. The focus is on the so called approximation algorithms previously in the last lecture. Or we talk about is exact average. We want to compute the solution, which will give you the best solution, best objective, right? Or optimal solution, right? What's the maximum you can get? Right? What's the smallest number can ever use to achieve some goal? Right? We are always going for the absolutely solution. But sometimes some problems are not so easy. You cannot have following the time algorithm, compute the best solution. When that happens, we have to resort to the so called approximation. The prostitution means the solution is, okay, all right, not best. But we can prove the solution is very close to the passage. This is the main idea of approximation. This slide is adapted from holy sites. He's aa young professor in a how do you ok they have the cost earlier than I do. So I make a reference to his size for two subjects approximation algorithm and online.

Now, this is also something that I would like to tell you about when you do something like force project or other works ok remember to give credit to the reference you use from other sources ok this is very important.

Now we start with a very easy problem called nasa, go to the next sector. So next sector, this work comes from aa famous, sorry about the some roberts and some lucky person. I yeah, so what location I used? Anyway, it's something like you enter a mysterious cave, right? And then bring two guys that the lucky person. It's not taking treasure from the cape, only the greedy one takes treasure from cape.

Anyway, let's assume you bring some bags. It's okay. And then there are treasure in the cave. You want to decide which items you put in your bag, right? To take it out. All right? Now your bag is a limited size. You have limited weights. So you cannot suffer too heavy hydrogens. Here, every item has a so called cost. So we put an item into the bag. It will curse and cost. And then there is a total cost limits of the bag. Then you cannot put in too many items. You can not to putting items to the very big cost ok so what's your goal? Is to maximize the profit or value? Are you take us? Right? This is one. That's the problem called next second problem. There are two versions of the problem. One is decision version. The decision version I tell you, you have the budget. B your items cannot exceed the budget. Then I will ask you, can you find a collection of items without exceeding the budget? But with the value higher than a threshold?

For example, my threshold that is 100. So i'm asking you, can you take items whose total value is larger than 100? Right? This is a conversion. And then there's also an optimization version. Or is that measurement is said, while you do not go beyond your budget, I want to make my value of the cf times as big as good. The difference of decision and composition is just decision in checking with the threshold, right? And the optimization is the training as it is possible. Right? So one example here are, sorry, four elements, right? A one, a two, a three, a one. And then they have a cost down and budget is full.

Now, we can have a solution taking item one and two, and then the total cost is four and total values. This is one possible solution. You can also take one and three. One of three, the cost is three, very small, and that is big like nine. Hopefully. This one seems very good, right? Then they are, can I do better? I you cannot do even better. So how about we take item one to answer? And then you will cost become six. Although your value is bigger, value is ten, but your cost becomes 66 is larger than budget four. This is not a value. That's not good. Right? Okay? Actually, the second one should be the best solution in the example, I think. All right. Now, the goal is to find a best solution, right?

Now, let's see how to use dynamic problems it is out of starting in unemployment. Now, here, dynamic programming usually have the following philosophy assess. Suppose I can see that only from item one to item five. Okay? What's the best I can do? Or can I achieve something? All right? And then when I have a choice of one item, I plus one level handed, right? I want to establish a recursion or recursive relation between smaller problem and a slightly bigger problem. All right. Ok so now on this page, you will see a lot of symbols, but don't worry, ok or I will explain. All right. Now, sip but it's sipsip is a subset of from items from one to I with the minimum cost and has value exactly p what does it mean? It actually means that we will want to achieve exactly the target key, exactly the target key. And then we want to keep the cost as small as possible.

Now you can only see that from items from 1 to 5, you cannot be on. Okay, on 1 to 5. On a cheaper car, some topic with the minimum cost. Why do I do that? Because I want to save more costs for future items, right? If I can do minimum cost now, then I would have a bit more budget in the future. So when I include which I can be, I will have an advantage. Sip exactly means this. All right? And then what's the cost of sip the cost of sip is easy. Now, when you do have such a subset, then you just add up all the cost of the elements in the subset, right? Is it just cost summation?

And then if this estimate does not exist, which means you cannot reach that party, it can not mention, then it just infinity. So because of this. Okay. Now, recursion, what is recursion? Well, at the very beginning, now remember about when we try to do impression, we always need to specifying the sum, right? Selling point means if I only see that from one item I equals one, then what happens? I equals one, right? I equals one, ok I equals one. That means you only have one item. Then what's your set? It depends on your topic b right? The value you want to reach, if the value you want to reach is equal to the value of the first item, then you are happy, right? You just said because I that's the best opinion.

In that case, you see this the item I selected, just the first item, right? Otherwise, it must be end, because otherwise, no matter whether you see that first item or not, you cannot cheat. You cannot reach that part about it, right? You cannot be exactly the part that no way then for si plus one p which means that now we have the solution for the smallest problem, how to build a solution for a larger problem. So in that case, we will need to specify some conditions like this one.

Now, the main idea is the newly added. You need to decide whether you be included in your solution or not, right? There are two choices. Either it loses your item, or you do not only like, right, only two choices. Now, where do you include the new item? There is a condition, right? Not every time you can include the new item. So here is the condition. B the b one. The new items value is less than equal to p new item is not too valuable. It is too valuable, including new item. It's already too much right beyond the very kind it is, not here.

Now, on the other hand, you should guarantee that if you reduce the value target by the value of the last item, you still have a solution from the first high ends.

That means cin and new highs will not hurt the feasibility of the whole thing, select a new item. And then you sell something in the old items, they can still be a proper valuable piece, right? Okay, so that is okay. And we make sure from I one to I they can reach a valuable p minus the value of the last item. With these two components together, then you can do recursion and say, then I have a choice. Either I see that the new item, right? I see that new item, and then take a solution from the earlier problem. Or I do not see any item. I just take a solution from earlier problem. So I have two choices. Remember, you got a second item, or I do not see that environment. Right? So these two choices, which one is better than it is english. Right? This is when the condition is satisfied.

Now, if the condition is not satisfied, then definitely you cannot include italy, right? And then I just take the solution from that earlier on the whole problem from 1 to 9. Okay? This is the whole thing, the recursion. And then you see the recursion. This is the sip thing. You need to compute sip for all the possible I and all possible p their combination, right? For example, I is one, p is one, right? I is two, p is three. So you need to consider all these possible combinations. Right? Remember, in the diamond, probably you say you need to worry about how many sub problems you need to compute. Right? Ok now here comes some problems. Do we have? Well, easy, because I is from one to n you have one, the first dimension, different values.

Now, how about second dimension? Pp is total value, total value. The largest total value is just a n times the largest value of any single item. Vp star is the largest value of any single item. Now you have n items. The largest total value will be n times v star. It was this, every item is nice, very much. I didn't do not add up. The total value will be single represent times n right? The largest single item value times n the total running time will be n times this.

Now, this is the solution that the final solution will get or not elaborate. We'll focus on the running time. Remember, the running time is what? N n times visa. It's n squared visa. Now, throughout the course, we will talk about the so called polynomial time running time. This one seems polynomial, right? It's n square as well as phenomenon visa. It's dna in design. Right? Anyway. It looks like a polynomial time running time, but it's not true. This one is not on the real time. What's the reason the reason is about the inside? How is v sounds right? So specific visa, what is that? Is the total value, right? If the ordinary side is the largest sector, largest value of any single item, the largest value could be 1,001 000. Then when this value 1,000 is input into the computer, and how many bits are needed? Do you need 1,000 bits? No. Right. In order to be put the value 1,000, you only need 10 minutes, ruling 10 minutes. So your input size is ten, not 1,000.

If your running time is linear in 1,000, it's not for a little time. Because now this 1,000 becomes two to the ten factor. So two to the ten, the emphasis is in export exponential position. This exponential time is not polynomial time. Therefore, what can be phenomenal? Only if the running time is log v star to power c and then this one we considered to be polynomial time. That's also why for the current one here, we call it pseudo polynomial. So it's a big thing. Okay, it's not true. Pseudo. All right? What pseudo is, right? Okay. Super fun. Yeah. So I guess I already finished explaining this is an exponential, right? This is the exponential algorithm. But we can transform this algorithm into a polynomial time. But we have to sacrifice something. What do we sacrifice? The so called ultimate? This is no longer optimal. It's only approximate solution. What do we do here? Before we analyze the the approximation algorithm, will first define what is the so called approximation ratio.

That means how close you are to the optimal solution. Here, nasdaq is a so called maximization problem for maximization problem. You are trying to make that as big as possible. So your algorithms that is not that big. It will be a bit slightly smaller than the optimal solution. When you do the comparison, you divide your solution by optimal solutions value. This value is always less than one, because you cannot be adapted, right? Optimal can be very big, but your solution is not that big. This division. This one is less than one for less than one value. If this ratio is always larger than equal to offer, no matter which, from instance, you have, it's always larger than equal to offer. Let me say it is an alpha approximation error. Ok this alpha can be half, can be 0 . 60 . 9990 k it could be a very smaller than one, right?

Now, let's see. We are trying to design a suit for peters for the next problem. It does has a name polynomial time approximation scheme. Let's see what is. Now, earlier, we are computing sip for all the p all the possible for that. Now, I say, because the total value this seems too big. I want to somehow skip something. Ok so here we are increasing like the one you said, but now I do not increase the one. I will increase the k right? So I tried anyone. Then I tried better k plus one, and then I tried better k plus two, k plus one, as if something that if you see, then the sub problems you need to compute becomes less, right? And then you already have to be polynomial. Ok so let's see. We will try to achieve some running time like this. N square, v star divided k because if you skip some values in between, right? Then the total number of possible values will be reduced by, right? It becomes one of the key fresh of all the possible banks.

Now we want this one to be polynomial. Okay? Now what do we do? Now we make episode a very small. Then we define k to be epsilon times v sub divided by n now, this value, it could be big, because we start to be super large, right? Even if epsilon is small, this one, the whole thing should still be big. Especially in our setting, k must be valued at one. Right? Here is a large average. If k is this one, then what's the yeah, okay. What's the scaling we do? Is, okay, is this? Because we will divide all that is by k what we did here is that we have a new value function, vi prime, which is the vi divided by k take a floor function, which means if it's 3 . 1, I just take three. If it's four and make it four, anyway, it's a value. The integer slightly below the value ok if we divide the viyk and round it down to be I prime, now we are dealing with the I prime.

We do a dynamic programming for this new value function, vi prime.

Now, what's the running time? The running time becomes what? Because k is this. If you put k into this formula, you will see that as power, actually, we have this, right? Absolutely star. And then we start canceled. It becomes n cube over epsilon. This one is polynomial time, right? Because it's just n to the three. Right? Because ii told you capsule is a small number, right? It's a fixed number, fixed number. So it's a constant. It does not affect the running time. Although you have a very small x zone, that means you have a long margin in time. But anyway, it's a constant. So it does not affect your running time scale. The scale is empty, free, right? And you can also see that if you want to, that's the next page.

So you want to get back approximation, then you need to spend longer money. Let's see how good the solution is, how good is this? Now let's see. Because we have a skilled value function. Now, for skilled value function, it's also a feasible nasa problem population, right? It's still an essence of nasa. In this new, smaller instance, you also have the best solution. Ok let's use s to represent the best solution when you consider the value function, the prime. Now, remember, this s could be different from the o always the optimal or best solution when your value function is b okay. It's only related to that account. What was the best solution to b and s is the best solution for being part.

Now, let's build up the connection between the two solutions test. And we want to found the value of s because s is the solution we computed. We want to use this solution to replace all. We have no way you can do all that always too difficult. We only compute s but I want to say, actually, the value of s is pretty close to value of. That's although, right? So why close? Let's see. Vs the value of s is larger than or equal to k times v prime s it is %. We try to build a link between value function b and value function b prime. Now, why is it true? That's easy, because the definition of b prime set up, b prime is b divided by k right? Ok and run down ok you see, if you put k over here as v divided by k right? Larger than equal to v prime, that's easier. That's just the definition of v prime. Now, secondly, v prime s actually is very good. Why? Because we said s is the best solution. For v prime, v prime, s must be larger or smaller than v five o very large, right? It must be larger.

Then we know, because the under valuation function be prime s is better than larger than k times prime b prime o now, next, we travel can be bounded from below by this one can be bounded below by vo minus KO but this is size of the solution. How many items you take? Why is that? Again, it's because of the round down operation will be fine. Let's see. You see this one. What is v prime? V prime is equal to v by VI divided by k round down. What's the round down? If you take round down, I recovered the original value - 1 must be even smaller than the round down. Right? You have a value here to say, round down and go down a bit, right? But what is - 1? - 1 is - 1 thing, one whole unit, right? It must be below your rundown background. That's why you see VI prime is at least VI minus overtake - 1.

This one, if you multiply k to the left side, it becomes kvi prime, which is here, right? Kd prime, right? Kd prime. Why don't you multiply k over here?

Now, on the right hand side becomes VI minus k right? Vi minus k and VI part becomes vook because Vivi becomes vo minus k becomes like it depends on how many items you have in the solution. Right? You have ten solutions, ten items, then it will be ten times k right? You have twenty, 20 items. It becomes 20 times k and anyway, it will be the number of items in your solution. O times k right? All right. Now, we are somewhat close to observe what they are doing. All right? We are somewhere close to be because we are trying to bound, we're trying to estimate vs right? Compared to the old.

So now we are a bit close, but still we have this. This is not good. Let's do a further relaxation. K times number of items o it's definitely larger than or equal to k this one minus kmn is border axis, right? The number of items also.

Then so if you minus a lot more sex, then the value comes even smaller, right? So this is correct. This is pretty easy. Isn't it? Right? Okay. Now, next, remember what k is. What is KK is defined as this, right? That sometimes these are over in the class. Kn is exactly equal to absolute visa, right? That's good. We have one more step close to the final goal. All right. So what's our final goal here? The final is trying to make the right hand side. This one, one minus epsilon op oobok ooptok it's called at we want to say our solution is larger than equal to one minus episode times the optimal solution. This is our goal, right? But then this goal and this one, you still need one last missing step. What's that missing step? If you want to have this, you have to show v star is less than or equal to ok right? I if v star is small, right? V star is small, then I replace epsilon v star by epsilon opt then the whole value becomes even smaller, right?

Because I make the stuff become opt so we hope that the size is small, is satisfy this relationship. If this is true, then we can have a final step that is right. Okay. Now, when you have this, then you can claim now my solution is the so called one minus epsilon approximation, because my solution divided by oet is larger than equal to one minus x then my solution is pretty good. Now, the major thing here is why besides less than okt now this one, actually, if I give you an arbitrary instance, it may not be true. But if it's not true, we can do pre processing to make it.

Now, let's see when will this not be true? The largest items that is larger than what would they have? It means that my optimal solution cannot take that, right? Because if I can take it, I will never let I have. The largest island value will never be liable. Beside of it just means that or we cannot take the item, right? Why? If we cannot take the item, why? The cost, the only possibility is the cost is too big bigger than budget. There is for that single item, the cost is too big, bigger than budget. Then what can you do? Then this is an induces, right? No matter which solution you cannot take it, right? Okay, they just do that. I change the instance such that there is no such idea. Agree. Now that's then our final step. You can just assume all the costs ci is definitely right. If there is somebody not in the budget, just get rid of it. You can never take it. Then that's the whole story about proving one minus epsilon approximation. I guess we'll take 10 minutes break and come back to the next example.