### Algorithms\_week3\_lecture2\_2-20240914

说话人1 00:00  
This part is will be passed. So what is ff is fully for the time approximate speed. Because in this running time, it is polynomial, involves n and one over x ok it's probably both variables. But this one, it's not only know of these two, because one of the epsilon is on the shoulder. This is even more unrealistic, because you can imagine it has almost 0 . 01. Then it's already into power 100. Right? It's just a not acceptable. This one we call pizzas. If you can separate n investment, one over epsilon like this, it is called happiness. Okay? Now i'm going to use about 10 minutes to review the concept of an enhance. How many of you have been in people are raising? Happy half, only very few.

Okay? Let me just start with the explanation of np if I show this strong, what is np now, before mp you have to know what is p right? P is easy. It means that it has a fundamental right? Ok so these are easy. Now, what does it mean? Something does not mean, not be. Okay. Remember this time it's not be okay. This man is not deterministic. What does it mean? It means that if I give you a solution to the problem, you can verify very easily whether this solution is, just like matt said, let's say if I give you a bunch of items, telling you this is the solution I give, you. Can you check whether this solution, the total value is above the threshold? Can you check? Yes, right. We can also check whether the total cost is the budget, right?

So basically, for the decision problem, decision version of on behalf of the next second, it is an np problem because I give you a solution, you can check whether budget is okay, whether that is okay, right? Okay, so send me to check. But for the optimization version of the next second, it's not true. I give you a set of items. Do you know whether this set has a larger standard possible? You don't know that if you always solve that problem. So even if I give you a solution, you do not know whether this is the best. So usually, for decision version of the problem is within it of alternation is not dna but then later on, i'm going to show you those are np continuous or np hard.

What is np completeness? Np completeness means all the problems in np all the problem in np they can be transformed into your problem. This is your problem. In other words, if you solve your problem, you can solve all the problems in np so in some sense, your problem is very difficult. Your problem is the most difficult problem. Important. Right? A this is called an incomplete ok this is np the data problem is called np complaints. Then this corresponds to the decision version of nasa.

Now, if you can show some problems, so if you can show all, the empty problems can become your problem. So actually, ii made one small mistake here. It's a critical mistake. So you have to also show your problem is within mpk if both are true, so all the empty problems, they can transform into your problem.

And your problem is also mp then we say that now, if this condition is missing, if this session missing, which means your problem, even if I give you a solution, you cannot verify. You cannot check whether the solution is correct. In that case, we say the problem is np hard, np hard problems. They may not be.

Okay, so they are even more. All right. Within and among the np hard problems, there are two different levels of harness. One is called strongly. The other is normal, okay? So what is strongly hard or aa completely? So let me just open this. What's the strong, strongest? Even if I give you as input in a very, not efficiently. For example, we have the next problem. Remember the value. Each item has value, so this item has value 1,000. Previously we say when I give you the equal 1,000, I ii only need to use 10 bits, right? Only time is necessary. Now I say I will give you 1,000 bits to input my value 1,000. Okay? In that case, the problem is still hard.

Then we say it's strongly now, that also means knapsack is not strongly because knapsack has a running time, has an algorithm whose running time is n squared, the stock. If you input v star, really, in the scale of v star, it's not hard, right? It becomes linear in v star. That's probably the time. If I give you equal, give you waste in the scale of the weight itself, then the nasa is among the difficult, right? Then we say nasa is not strongly empty. Part nasa is only normal. All right. This is a difference between strong and normal.

Now, let me find that way. I told you before, right? Then sometime later in this lecture, i'm going to show you how to prove mp and behind us. But now, now I guess I would not explain the procedure, because there's no concrete example. And for this part, I already explained to you the difference. All right. The example here, I guess I also skip. This is only one possible example to show them. The problem is to be hard. As I promised, I spent less than 10 minutes to cover something which should be covered within maybe 2 hours lecture and go on to the study. But because we need to cover a lot more serious. So for this part, I just cover to this exam, right? In the quiz for final exam, I will not test you how to prove it as much its undergrad. Now, we will seriously look at some other interesting graph problems. They will help you understand better the approximation now and also from now on. We pay special attention because there will be some bonus points arising.

Now, the first ground problem we talked about is called the tax cut. What is vertex cut up? And you show your figure in the graph. This is a graph. And the graph basically consist of nodes and edges. Those circles are nodes. The lines edges. If every edge connect two notes or two vertices. For this picture, vertex server says, I want to see that a set of vertices, such as every edge is touched by everything, one vertex you choose.

Then when I show you this solution, can anyone tell me whether the black nodes are the solution or the white nodes are solution? Every edge has to touch at least one vertex. You chose. Very good white. Okay? The white point that turned up here, the white one is vertex color. Now, because every edge, right, you at least have one very white one, right? Every such edge, at least one wider. So see, that advice will be a valid vertex coverage. Okay? Basically, vertex cover means using vertex to cover all the areas. All right. Okay. Good. Now you also have a optimization version of the vertex. We want to find a vertex server whose first is few as possible. Okay? Right? Many numbers. You can see that three to come all good. You never set forth. Right? Now, this problem is going to be hard, although im not going to prove why is it right? We have to go for the approximation area.

Now, this one is different from the next second, because this one, we are trying to minimize the number of vertices you chose, right? It's a minimization problem. We still do division like your solutions quality, divide by optimal solutions, quality. Ok this division of this ratio, it is less than or equal to offer for every possible instance. Then we say it is an offer approximation.

Now, here, for minimization problem, this model always divided. But you have been worse than normal. You have to get more as a side. Compare optimal search, ok so now I see what can we do for what excellent. But one very straightforward idea is, I will see that one of the text, which can cover some edges, which are not covered yet.

Ok this is a very straightforward idea, but which vertex do you set? How to set them up? Can cover a lot of exercise, right? So this is some kind of greedy algorithm, right? I want to see that number of tests, which can cover a lot of edge, right? Or I can cover the most number of edges, then this physics must be very valuable, right? Good. So this is our second idea. Pick up the text, what you call them most relevant. This is, in this instance. What's the algorithm there? You see? The first note we take is this one, right? This one, if you pick it, you can cover some edges, right? After picking this one, then the other edges are just the same. This is 13 independent edges. And then you can see that one from each edge, then that would be a solution, the greedy solution, ok greedy solution. You take four notes. What was the optimal solution here? Three, right? You just take the middle three. Where is this taking part of all the edges? Right?

Good. Now, but this method is not, but one thing is not optimal for sure, because you only see the optimal three. And then the other one is only it's boring. Then it can be even worse. So the figure on the right it shows how bad it can be. Now we have two layers of notes on the top layer. You have n nodes and vertices. In the lower layer, you have different colors at each different color. They need different things. For example, r six, r six means this node connects to all the nodes. Ok now here we are, using example, n equal to six, okn equal to six. That's why we call r six. Basically, this blue note connects to every node on the first day. And then r five as well. It connects to five notes on the above 5 minutes. Four, I connected four notes. Three is the first different thing. So three will cover. We'll connect to three nodes on the top layer, and they have another pass remote connecting to another, ask three nodes on the upper layer.

So basically, the construction looks like this. Every ri has so many nodes, n over I rounding down, basically trying to cover a certain number of nodes on the upper level. And then you need to have multiple copies, if you will, have to right cover as many as possible the upper level notes.

So you see, here we have two r three nodes. And then let me see three r two nodes, right? You have three arguments because afternoons can only cover two nodes. And above, when you have support together six, so you have to cover 22, and 2325. So you have to maintain three partners. So by doing this, which node has a largest degree, wait. Sorry, I didn't tell you what is 3 years, but degree is the number of neighbors, or it is not connected.

Then our is not there, our is actually virtual, nothing, ours is nothing. Yeah, how much should be done? So looking at all the colors here, all right? And all the connections here, which one is the largest. So actually, I want you can also say, ii have some individuals they come connecting this, but anyway, the largest degree one, what candidates are six are 67 ° 6. All the above one has to be five. These are 35432. So they are not as big as six r six is the largest difference, largest neighbor, largest number of names, ok so your reading method will take r six first. And after taking r six because blue lines are long. So now the upper layer has to be for, at most, right? And then your outside becomes the largest degree.

Ok so by something, then the reality will take us first. And then they are fine and then they are four, right? And then take r three. There are two. Until I take all the animals, I then I want you to be anyone as well because there are no extra remaining, right? If I take all the notes in the second layer, then no actual show up, right? They are all covered. So your solution takes all the color notes. Now, how many kind of bills do you have? Let me see it should be. So actually, I should somehow wrote here ok so how many colors do you have? You need to say is n over. I take a running down, right? So this I can be 12345, right? Okay? So it will be roughly over one.

Now i'm cheating a bit, right? Because i'm cheating that r one also connects to the upper layer. R one has altogether n nodes. And they are connecting to those upper layer, but it's okay. It's just an issue of breaking time. So you have this. This is how many notes you have in the bottom layer.

Now, for this one, you take all the end out and then will be a common term. And inside is 1 plus, half plus. What do we call it the whole sequence here? What's the name of this? 1 + half + third + 4th. Did you? Yeah, so in chinese, I know you put your name. So in english is called harmonic sequence, right? Harmonic sequence so that the sound of harmonic sequence is. How much? Log n this is sum of harmonic series altogether. You need to see that n log n notes as a vertex cover. What's the optimal solution here? What's the optimal solution? For this instance? You can guess, right? You already select the bottom layer, right? Then also you can set up the top layer, because you only have edges between the top layer and bottom layer, right? If I select all the nodes in top layer, then I can also put on the edges, but you hope they don't have animals. That was better than m again. Here. What's the ratio is? Log n if you compute the approximation of this vd algorithm is log n approximation.

Ok. Right? That means this reading method cannot be better than log n approximation. That's not good. If how good approximation, the vertex still can be ok let's see how the ok another algorithm case. What's this area? That is very easy? Says, now I take an arbitrary edge, arbitrary action, and then usually only need to take one and five, right? To cover this edge. And now I say i'm being very generous. I'm taking two false, both end points of the selective action.

Then I can move all the edges that can be covered by these two versus then the second equation is in the same. Again, I pick an uncovered edge, take both inputs, and then this is a very simple error. How good is this time? And actually, if you write it in suitable, it's also very crazy, right? Take an edge, take both end bonds, and then remove everything. They have any connection to these two vertices.

Okay? How good is this element? I I thought, to explain a bit ok so basically, your algorithm takes this happened. The things are so called matching. So if they are able to go on for 4 rounds, then the edges they have to pick must be independent. They cannot have common vertex. That's not possible. For example, you cannot have this node same as this is not possible, because when you pick the two nodes here, it will remove the elbows and remove anything. They have connection with this. Anyone, right? The second time you take an edge, the end points cannot be the same as the end point of the first edge.

Basically, you pick an edge, two nodes, and then everything that have connection to these two nodes I want.

Okay? And then the next action pick must be isolated from the first edge. They have no connection at all. Your elbows will take something like this, a bunch of independent edges. Any questions? Everything will never say. I take one edge here, and then ii take another edge, which is somehow like this. That's impossible, right? Because this edge is already gone when you see that this mode. Right? Good. Now this is what you have. Then that means your algorithm will select two k nodes, right? You are looked at two cains. How about usage? You don't know how many quite optimal situation, but you don't want to say optimal solution has to cover all these edges, right? That means an optimal solution for every edge here. It needs to see that at least one vertex, right? Otherwise, they actually not become optimal solution. Will say that how many edges, how many nodes only see what you want? This is this lalg will be equal to a right? What is optopt is larger than is that right? Because for all these key edges, open d has to select one node from each of them, right?

Okay. Now it comes very obvious. Opt at least kl was the two k you divide the algorithm by opt it becomes what? Less than two, less than equal to two. The approximation ratio of this algorithm. It's true twice the optimal solution. Twice. What is it? Very simple errors? And you can get a two approximation, right? I guess I will not be too go through with this, just too easy, right? That's just too easy. Fine. Good news. I can do the two approximation for the textile. Any questions about this? Too? Iii think that this too should be a relatively easy to understand approximation in all the topics you are going. Right. Now, it's a be a good point, which is to say, how about we add weight on top of the notes of risks?

So now, for this same instance, probably have wait one for the yellow vertex and weight three for the black vertex. I want to see that a minimum weight collection of notes to cover the areas. Same story. The only difference is I am waiting, right? When i'm waiting, it seems that you can no longer do tricks like the algorithm. You just say, take an edge or those points. It seems not working, right? So then what can we do? Then we can use the knowledge of so called integer linear program to dominate the problem. We use variable xi to represent whether I see that node either or not. Then your vertex server, this corresponds to an assignment of the xi value. If you assign xi to be one, that's there is no, right? There is no, it's zero. I do not see that. Then your objective function becomes the summation of weight of no line multiplying xi that means if you see that no ixi will be one, and this one will be multiplied double line. And this number I will show up in the objective function, right? You consume such weight.

Then the constraint is easy. Constraint says for every edge, you have to take one end point of the section, right? How do you specify that constraint?

Now, it's the first time on might be the first time you you see the same interview program, I will show you the constraint is very, very smart. It's just the next time class actually larger than you remember. Remember, xi is 01, right? If this constraint is satisfied, that means xifg at least one of them is one. If both are zero, this constrain cannot be satisfied. At least one of the xi dot xj is one. That corresponds to what at least one end point is selective for every edge. Right? So this country basically applies for every edge. As long as you have connection between two nodes, you have such a constraint. So overall, the population looks like this. Very easy formulation, right? Minimization of the summation of weighted sum of xi subject to this, st means subject to these constraints. This ij is an edge. Now, any questions?

Now, they're soon entering the first in class exercise. Pay attention to them. Now, what's the general format of individual program? It looks like this. A bit more general minimizing the weighted sum of xj subject two is constrained. Then xj logging into zero, then xj is integral ok this is a general format. Usually, if you have learned the mathematical also integer or linear programming, before that, actually, this constraint can be written as a matrix matrix a times x larger than equal to a vector b but again, we are not doing this in the class of the worry about formality of the in the program.

But there is the good news is, because vertex cover is npr what can we conclude? We can conclude integer linear programming is np hard in general ok remember, my discussion ok into linear programming problem is np hard, in general. But very often, if someone do not learn algorithms, then they claim the following things.

I have a problem. This problem, I can formulate it into an intelligent program, okay? Then they say then this problem must be going to be hard. Is it correct? Something can be formative as one specific infinity program, right? Then because it is important is enough. So my problem is on behalf. This is very dangerous. This is actually my first in class exercise for the first half of semester. Can you find one problem, which is easy, meaning you can find a little time error. Okay? But at the same time, you can formulate your problem into an integer program formation. All right. Now clear about the question. Find one easy problem. Formulate that problem. The problem can be formulated into an integer linear programming formation, something like this, some special c some specialty, and some specialty.

Clear? All right. You're saying, but indigenous program is hard, in general, how to solve it. People usually relax the integer integral constraint to believe. In other words, it was the relaxation of intervening program to be linear program. Everything is the same, except they remove this last constraint. Integral constraint gone xi can be fraction ok and 0 . 730 k when that is actually, there are a lot of methods, they can solve the problem starting from simplex, which is exponential. But very quickly practice, you also have an eclipse or method, which is for another time. So but then you also have interior . method, which is for another time and also fast, right?

Now. These are, again, not our focus. You will probably use it very long I in the later phase of the study or work. That end. All right. So for waiting, what's the several problems? After we have this intervening program population? What can we do? So when we first relax this one to be convenience of the constraint xi value equals zero, no indicators.

And then we solve this to the optimality. We solve this linear program and against the so called optimal solution. X star, remember, this x star can be fraction and zero by something, but this is not a valid solution. Because, for example, you have a triangle, then I assign every exercise in 1/2. Then the constraint xi was actually larger than v one. It holds for every abstract half plus half, and so on. Half plus half is nice. But this is the final of us to select the solution, because half for every note, what does it mean? I don't know. Right? It's not a valid solution. Now, we are going to do some trick. First of all, the x star solution, you said that you compute it here. It is a lower belt of the actual optimal solution for the integer linear program. In other words, because you have less constraints, you see, compared to the model here, if I relate to this, then it means I have less constraint on xi then the objective value can only be better, right?

I have larger searching space, right? And then your objective can only become better. So the achieved value here, obtlp must be better than the original optimal solution for the integer, meaning. I guess you agree, right? Less constraint than better solution. Okay? What's the next one? Is? We're going to use this x stop to derive a famine intervals, which means ii need to have a solution which really seen as somebody, right? Really seen as somebody. What should I do? Actually, you can have a very good guess. Usually when you see fractional number, how to change its integer, the very natural method is to do routing maps when you are half and less than half, do that. Ok for half people upgrade one map, and for 0 . 49, they downgrade to zero.

We do this, for example, for the triangle here, then it means every vertex will be selected, right? Because each of them is half, right? So it's not going. Every half is one. This running solution actually is very good. Let's see how good it is. All right? We see that all the node I whose xixi star value is at least half, I see that into my search, I have a message, but this is the rounding solution.

Now, let's see how good this solution is compared to the optimal solution. What's the optimal? What's the solution you get? The solution you get? Is the s right? S ws is the weight of your solution. You add up all the wi on that side. Now, first of all, you will know that this solution is a valid vertex. Come on. What is that? If I upgrade every half or above to one, then it still satisfy the constraints by the name. Why is that? That one is actually pretty easy. Why? Because you see the constraint looks like that, right?

Then if some constraint is satisfied, that means these two variables, at least one of them is larger than or equal to half. Right? It cannot be the case that both of them are smaller than half. That is not possible. They add up to be larger than one. Right?

If some two values add up to the one or above, then at least one of them is that is half, right?

So if that one is at least half, so that one after running will be selected message. Right? So that becomes one, and then your constraint is still satisfied, right? Somebody becomes one. The constraint is still good. So all the constraints they are still satisfied. Right? Ok what's one thing? S is bad. Now, next thing, next time, weight of s is pretty good. In fact, it's very small because we want to find the minimum weight vertex. Why is it small? Let's see. We are trying to show ws is less than or equal to two times the optimal linear production. Right? How to show it?

Now, first of all, leading for medium program solution, it's adding all the notes, their ssr multiplying wr right? Now, this is your optimal, right? This is optimal. If I remove some vertex, I say I have, and for those projects, I do not assign them any better. Then the value can only become smaller, right? They can only transport. That's why they are. Then if I restrict the if I restrict the domain to be within s then the way it is even smaller. Now, this part, because every xix is not the s it satisfied this condition, right? Because I only see that those exercise samples that is not at least half, right? Xisi is larger than the half. Is that right? Ok larger than it is half?

Then what is this? This is douglas. You see that this douglas is a waiter. You said s the weight of the set, as you multiply this two over to this side, it becomes two times odt odt right? This is odt in the end, we have the relation basically show that the weight of ws is less than equal to two times is opt now, on the other hand, remember, optimal solution of linear program is less than or equal to optimal solution for the integer linear program. Okay. That's true or more. So your solution selected will be, at most, twice the optimization. Right? That is the proof of the two approximation, even for weighted the test company, right?

This is another way to show the two approximation. Now. Any question? The main idea is just you upgrade the value to one, but the value upgrade, they already reached half, right? They are already half the upward operation. You do just double the value. You will never be more, right? Is only double. So that's the russian effort. And then we have talked about the prosecution of two from above, right? And then people also show that you cannot do very good approximation for living. So the especially if p is not equal to mp you cannot do better than 1 . 3646 approximation.

Now, if you assume an even stronger assumption like, you need a conjecture. If you assume that one is true, then you cannot do better than one, two, minus s approximation. You cannot do this. Two minus epsilon is very close to two. Basically, it tells us to a population probably is the best game. And then we already have two measures, achieving into approximation, right? At least we did only one method, right? But i'm waiting. We already have two methods achieving to our partition. So that's the best we do. This is also some . you can use to persuade your future boss that your designs as possible. Nobody can be there, right? Because there are no bounds, right? No one can be there. Then it was. Yes, right? So next one, we'll talk about the very interesting set up, right? What is set up? Second is also easy. Let me show you an example. So we have a bunch of elements to these black dots. They are elements. Okay? You have a set of some subset of elements like the s one covering six elements, s two covering four, s six covering three, and three, s three, s four, s five covering four elements.

These s there are elements, their activity. What's your job? Your job is to pick some s to cover all the evidence. Okay? They would say they have, all right. I think more I can come over there.

Now you have to use the minimum number of sets subjects to cover audience. All right. Ok so that's the so called mean set card, right? Now, for being set cover, here, in this instance, the best solution is this. You choose three vertical subsets, right? You can cover all the elements, good. But it's not difficult. It was not easy to find these three. All right? It was not easy. But I usually use greedy algorithm. What's the very, most natural greedy algorithm you use here? When you see that as a substance? What's the criteria? For example, 1234567. If you see that one first, which one is that first? I heard one, iy one. Yeah, you can come the most points, right? One can have a 6 points. That's very good. Other people can only come for three to few, right? One can cover the most number of points. Let me see that one first.

Now, again, it's a review of reading out of them. Every time you pick the seemingly best option, right? A lot of things like most number of points. That's a check an example. So you choose s one first, right? And then after to the s one, you will remove all the points from the best one. And then in the remaining success, you pick somebody, right? And then which one do you? 66, very good, because six can cover no 3 points, but all the other things for 2 months, ok pick six. And then I have to pick six. The next one you pick is two. Very good, right? Because now two can cover 2 points, but all the others don't cover one point, right? Then you have to choose two. And finally, you choose. So you see, by doing this, you said before success algorithm is four, optimal three, the ratio is one, 4 ÷ 3. This is only for the specific example, right? It's not for the general case, so hope that this really helps and can be will set up a problem.

Well, let's see. Actually, the ratio is long. And right? That's a very valuation, right? Because remember, when we call a little test out, let's say I need a you can only achieve log in at a time. I told you it's not good, right? Because they don't have two options. But this one, I don't know. It's also, again, I maybe it's not good, but you will know you very soon whether it's good or not. But anyway, let's figure out whether we can prove this log in approximation. Want to show it from the positive cyprus. It will not go beyond. All right. For the proof logic actually is very, very interesting. Now, let's see. What's the proof logic? The truth logic is the following. Suppose possible solution only cxk subjects, okok now, your greedy error, what does it do? Every time it takes the best subset, right? They can cover the most number of points, remaining points for them. Now, actually, this is very good. Why? Because and you draw it again, because you can imagine optimal, say, seven k success, a couple of ones, ok this is ak subset, because all the ones right.

Now, let's see, the first iteration, greedy, select another one, maybe not. Some anyone here. Greedy does not select anyone here. Is there another one? Say this? This is greedy for this week. But also conclusions are optimal, select k subsets, cover all the points. That means one of the candidates in optimal selection can cover at least one over k minus one over kn points. You see, I draw five sets here. If it's five, at least one of the sub sets can cover five. One of the five n parts. Is that right? Optimal set five can cover all the points. What's whole point? All points is n points. The only set of five subjects can cover any points that there must be one set in optimal selection. That's probably at least one to the five end points. Is that right? The now what's this one? Is really trust that. So the number of points really choice can cover can only be better than any one of the five. It's optimal solution. Is that right? Ok so that means this greedy solution. It also can cover at least what will start in, right?

Okay.

So now in the first situation, really see that one subset, this subset can cover at least one over k fraction of the points, right?

With this after this round, how many points are remaining? Only this number of points, n times one minus one over k because one will locate fraction, is already covered by your first set in a greedy solution. Right? Only so much remaining. This is analysis for the first round. How about second round? Second round, the story didn't change. Always some points i'm missing back, some points when I got you have remaining points in the plane, right? For the remaining points, these five optimal subset can still cover them. All right? Is it right? Ok and then you're selecting a greedy choice, which is the best at the current moment, because it's best. So it can only be better than all than any one of the five. The second round selection by the greedy can still cover at least one of the five fraction of the remaining points. Is that right? Good. Then you have one more step. So now, after the second round, you only have so many points left. You multiply another one minus smaller k because another one working fractions on, right? Okay? Then it's become very easy. It just continue. After t rounds, I will only have so many points left.

All the only thing I need to make guarantee is that I want to choose some t such that this value is less than one. If it's less than one, that means no points that, right? Then you can take t to be k log n and then using some mathematics calculation, right? You will see this one is less than n times one over e log n then this is equal to one, which means this type is less than. All right? Done. You show that after log n after k log n max, you cover all the points. You will cover all the points. What's the optimal solution? Optimal solution is ki remember you use this value divided by k it becomes one log n then you show really algorithm. It can achieve log n approximation. Right? Ok next.

Basically, we finished the proof of the log n now, for second, log n is actually very good, because fighter shoulder that unless some very unlikely happening event happens, this log n is the best possible you can achieve. Basically, you cannot do better than one minus epsilon times log n approximation for sector. In other words, it tells you greedy algorithm is the best algorithm. What's that? Although the risk is not good. So I think I will take 10 minutes break, and they'll come back to the remaining. Maybe.