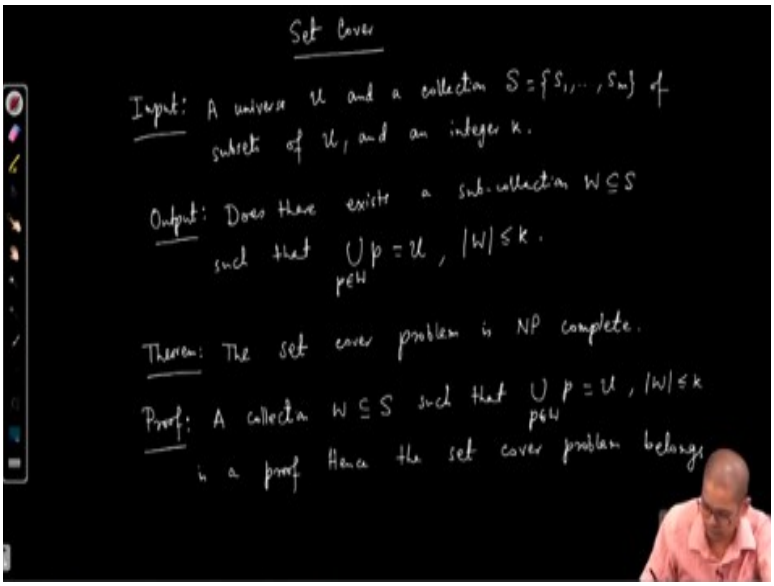
Selected Topics in Algorithm
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 38
NP – Completeness of Set Cover, Weak and Strong NP - Completeness

Welcome. So, the last couple of lectures it doing NP hardness and seeing various kinds of reductions. Today we will concluding with our last reduction we have shown a hardness of formula satisfiability problem and then hardness of various graph problems and problems with numbers. So, today we will show hardness of some set reduction problems.

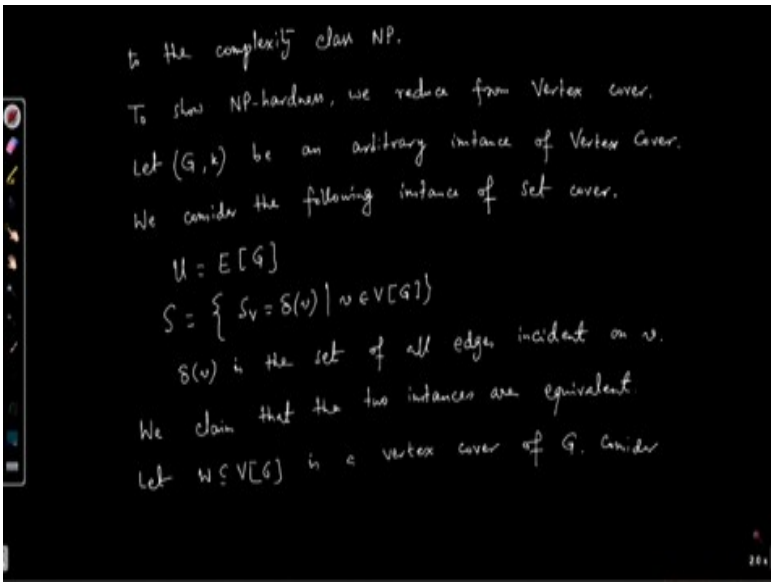
(Refer Slide Time: 00:56)



So, for that we take the set cover problem. So, what is the input? Input is a universe U and a collection is equal to $S_1, S_2, \ldots, S_m$ are subset of U and integer k. Output, does there exist of sub collection W and subset of it such that union of set in w is the universe and size of w is less than equal to k. So, that means whether the universe can be covered with at most k sets of the collection. So, the set cover problem is NP complete.

Proof, again the membership in NP is immediate because a solution a certificate could be a collection of at most k sets and it is easy to verify for their union is U or not. So, collection W subset of S such that $\cup_{p \in W} p = U$ and cardinality W is given equal to k is a proof hence the set cover problem belongs to the complexity class NP.

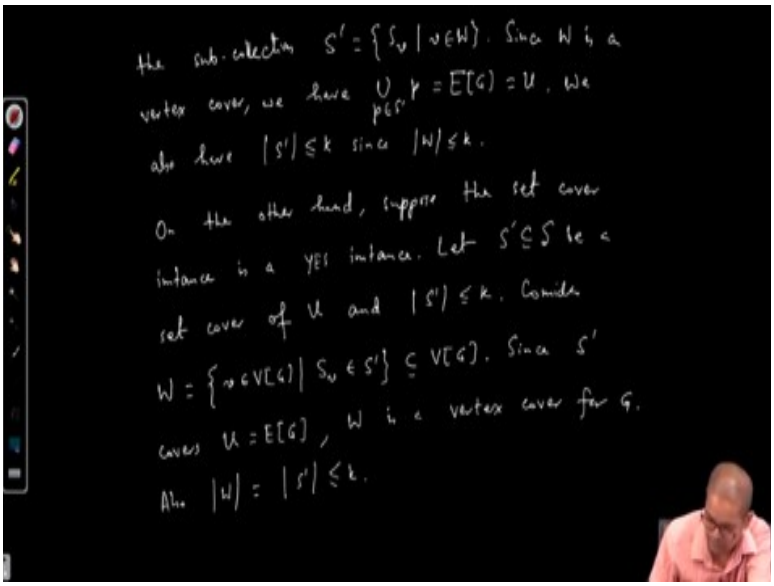
(Refer Slide Time: 05:21)



Now to show NP hardness we reduce from vertex cover. Again, polynomial time we need to one cover reduction. So, let G, k be an arbitrary instance of vertex cover we consider the following instance all set cover. Universe u is the set of edges is the edge set of G and the collection is that S called to S_v for each vertex I have the set or which is a subset of u and that is $\delta(v)$. $\delta(v)$ is the set of all edges incident on v.

Now we claim that the two instances are equivalent. So, let W subset of V of G is vertex cover of G.

(Refer Slide Time: 09:30)

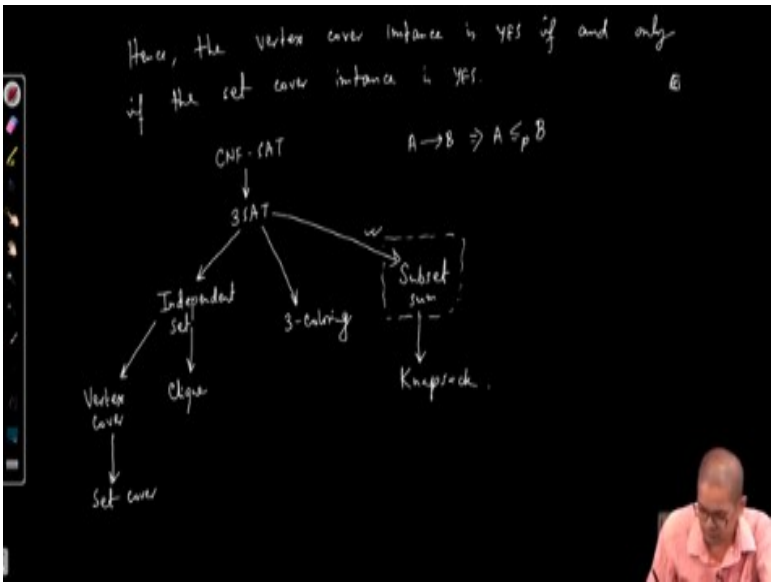


Consider the sub collection $S_1, \ldots, S_v$ such that v belongs to W. So, those sets correspond to the vertices in W. Now because each edge is covered or each edge at least one endpoint is in W is cleared that that loop S' covers U so since W is a vertex cover, we have $\cup_{p \in S'} p = U$ equal to set of all edges which is universe and we also have cardinality s prime less than equal to k in cardinality W is less than equal to k.

So, the set cover instance is a YES instance. On the other hand, suppose the set cover instance is YES instance the way to show that the particular instance that is YES instance. So, let this time subset of is set cover of U and cardinality s prime is at most k. So, NP if we pick the corresponding vertices that will form a vertex cover. So, consider W which is be in V in G such that they solve the belongs to the S'.

See the subset of V G this S' covers U which is equal to edge set of G, W is a vertex cover for G. And also, cardinality W is equal to cardinality s prime which is less than equal to k.

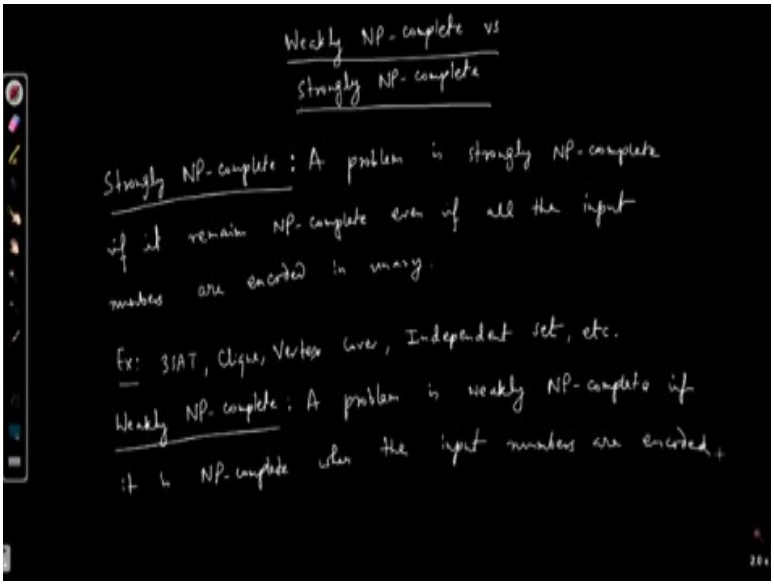
(Refer Slide Time: 13:29)



Hence the vertex cover instance is YES if and only if the set covered instance is YES. This conclusion. So, to summarize; what are the reductions we have seen in NP completeness lectures. So, we started with the top that the CNF-SAT is NP complete and we reduced CNF-SAT to 3SAT. So, the notation is A arrow B means A many to one Karp reduces to B. Then for 3SAT we have shown we have reduced 3SAT to independent SAT.

The reduced independent SAT to clique then we have reduce independence SAT to vertex cover we have reduced vertex cover to 3SAT, we have reduced 3SAT to same colour ability or 3 colouring of graph and we have also reduced 3 SAT to opposite from of we are reduced subsets of knapsack. So, this reduction needs special attention you know this invert number and the number you know we have represented in the number binary and this number is huge.

(Refer Slide Time: 16:58)

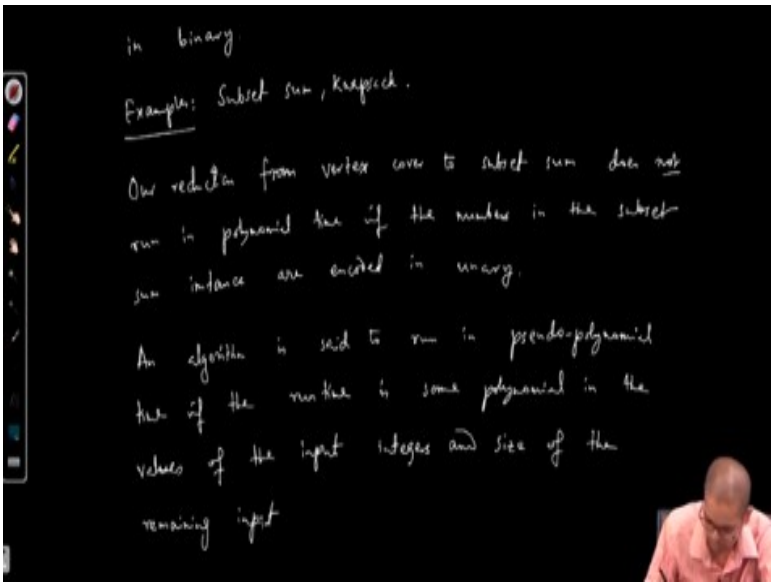


So, if something called weakly NP complete versus strongly NP complete. So, what is it? So, strongly NP complete a problem is strongly NP complete if it remains into NP complete even if all the input numbers are encoded in unary. Examples of such problems this is the problems that we have been forward are strongly NP complete are 3SAT and strongly independent and the 3SAT is strongly complete because it does not involve some number.

But the clique or independent set the input is a graph and the integer. So, even if that it is encoded all in unary that means to write k as input and write k 1. So, the input length becomes the proportional to increases proportional to the value of the numbers. They are also it is NP complete. So, the example of strongly NP complete; problems are 3SAT because it does not involve any integer.

Then all the typical graph problem that you have seen clique, vertex cover, independence set etcetera. Another hand weakly NP complete if the numbers are encoded binary. So, weakly NP complete problem it is NP complete when the input number are encoded in binary.

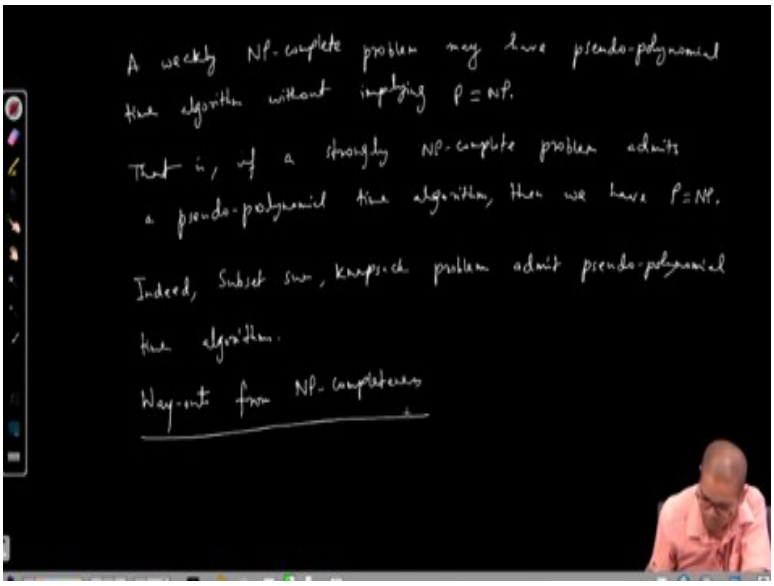
(Refer Slide Time: 21:46)



Examples, the subsets sum. We recall we have reduced from vertex cover in the numbers are huge and if the numbers are written in base 4 then it is representation is polynomial of the size of the instance of vertex cover. But it is the number has to be written in union then it is not a polynomial type reduction any more this subset sum knapsack. So, our reduction from vertex cover to subset sum does not run in polynomial time if the numbers in the subset sum instance are encoded in unary.

And hence we show only weakly NP complete and for weakly NP complete problems it can admit it is perfectly alright to admit pseudo polynomial time algorithm without proving P = NP. So, an algorithm is the said to run in pseudo polynomial time is the run time is some polynomial in the value of the input integers and size of the remaining input.

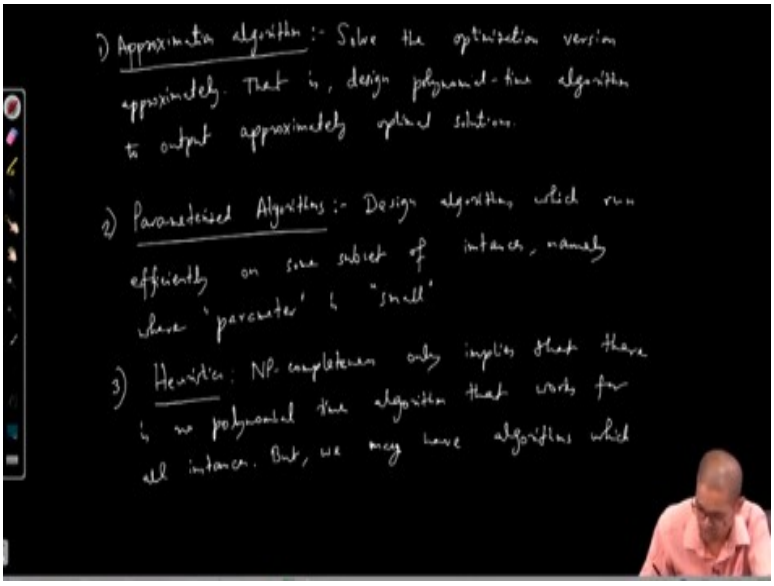
(Refer Slide Time: 26:07)



So, a weakly NP complete problem may have pseudo polynomial time algorithm without implying P = NP. That is if a strongly polynomial time, a strongly NP complete problem admits a pseudo polynomial kind algorithm, then we have P = NP. So, it is unlikely for a strongly NP complete problem to admit pseudo polynomial time algorithm. And indeed, subsets some knapsack problems admit pseudo polynomial time algorithm.

So, now next what? So, suppose we think that some problems with NP complete. So, what are the way outs from NP completeness. So, let us discuss some of the way outs.

(Refer Slide Time: 30:04)

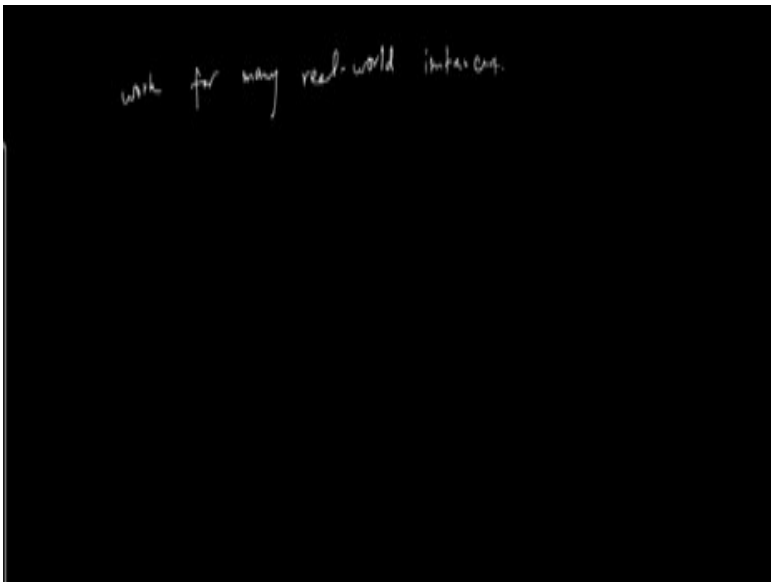


The first most well studied way out is approximation algorithm. A problem to NP complete means that the decision version of NP complete and that is same are saying that you know the optimization version also would not expect to have a polynomial time algorithm. Because if we; can solve optimization version efficiently then we can solve the machine version also efficiently. But the idea here is that solve the optimization version approximately.

That is design polynomial time algorithm to output approximately optimal solutions. This is the topic that which is that we will see the next couple of lectures in detail that is approximation algorithms. The second one is parameterized algorithms, design algorithms which run efficiently on some subset of instances, namely where the parameter is small. These also are success story for tackling entity NP completeness hardness.

And this also will be in great detail in that course and last and not the least in the heuristics. NP completeness only implies that there is no polynomial time algorithm that works for all instances.

(Refer Slide Time: 35:20)



But we may have algorithms which work for many the real world instances. This is what this curiosity design expressed the design algorithms which may not work for all instances but what will for real vertices. These are the three approaches. This approach not be able to see when you

are able to cover in this course. But we will see other approaches namely approximation algorithm and parametrizes algorithm in the remaining part of the course. Thank you.