Discussion 10

1. Given the SAT problem from lecture for a Boolean expression in Conjunctive Normal Form with any number of clauses and any number of literals in each clause. For example,

$$(X_1 \vee \neg X_3) \wedge (X_1 \vee \neg X_2 \vee X_4 \vee X_5) \wedge \ldots$$

Prove that SAT is polynomial time reducible to the 3-SAT problem (in which each clause contains at most 3 literals.)

Solution: We will turn each clause of size k in the SAT problem into one or more clauses of size 3 as follows:

Clause size	SAT	3SAT
1	(X_1)	$(X_1 \vee X_1 \vee X_1)$
2	$(X_1 \vee X_2)$	$(X_1 \vee X2 \vee X_1)$
3	$(X_1 \vee X2 \vee X_3)$	$(X_1 \vee X2 \vee X_3)$
4	$(X_1 \vee X2 \vee X_3 \vee X_4)$	$(X_1 \vee X2 \vee S_1) \wedge (\neg S_1 \vee X_3 \vee X_4)$
5	$(X_1 \vee X2 \vee X_3 \vee X_4 \vee X_5)$	$(X_1 \vee X2 \vee S_1) \wedge (\neg S_1 \vee X_3 \vee S_2) \wedge (\neg S_2 \vee X_4 \vee X_5)$
…		

In general, we can turn any clause of size k>3 into k-2 clauses of size 3 using k-3 dummy variables to "chain" the clauses together using conjunction as shown in above examples.

Proof: We need to show that the clause of size k is satisfiable iff the chain of clauses of size 3 is satisfiable. To show this:
A – if we have a satisfying truth assignment in the clause of size k, we can find a satisfying truth assignment in the chain of clauses of size 3, because the satisfying truth assignment in the clause of size k requires at least one of the terms in the clause to be true. This will cause one of the (k-2) clauses of size 3 in the chain to evaluate to 1, we can then use the k-3 dummy variables to set the remaining clauses to be true and therefore find a satisfying truth assignment for the chain.

B – if we have a satisfying truth assignment in the chain of clauses of size 3, it must be that at least one of the X_i variables is true (because the dummy variables on their own can only set k-3 clauses to true). And a satisfying truth assignment in the clause of size k requires only one of the X_i variable to be true. So this will be a satisfying truth assignment.

2. The *Set Packing* problem is as follows. We are given *m* sets $S_1, S_2, \dots S_m$ and an integer *k*. Our goal is to select *k* of the *m* sets such that no selected pair have any elements in common. Prove that this problem is **NP**-complete.

Solution:

1- Prove that Set Packing is in NP

Certificate: a subset of k sets (out of the m sets given) that have no elements in common

Certifier: Can easily check in polynomial time that

a- There are k sets in the certificate

b- The k sets have no elements in common

A and b can be easily done in polynomial time. → Set Packing ϵ NP

2- Choose independent set for our reduction

3- Will show that Indep. Set $\leq_p$ Set Packing

We will start with an instance of the Indep Set problem (Is there an indep set of size at least k in G) and will construct a set of sets such that there are k of them that have no elements in common iff we have an independent set of size k in G.

Construction of sets: For each node i in G we will create a set S_i. The elements of S_i will consist of the edges incident on i in G.

Proof of correctness for the reduction step:

A – If we have an indep set of size k in G, we can use that to find k sets that have no common elements. The reason is that since the k nodes in G are independent they do not share any edges, therefore the sets corresponding to these k nodes will not have any elements in common since these elements correspond to the edges incident on the k nodes in G.

B – If we have k sets that have no elements in common, we can find an indep set of size k in G. The reason is that since these sets do not have any elements in common, the corresponding nodes in G will have no edges in common or in other words they will be independent, and will form an indep set of size k.

3. The *Steiner Tree* problem is as follows. Given an undirected graph *G=(V,E)* with nonnegative edge costs and whose vertices are partitioned into two sets, *R* and *S*, find a tree $T \subseteq G$ such that for every *v* in *R*, *v* is in *T* with total cost at most *C*. That is, the tree that contains every vertex in *R* (and possibly some in *S*) with a total edge cost of at most *C*.
Prove that this problem is **NP**-complete.

Solution:

1- Prove that the Steiner Tree Problem is in NP

Certificate: a tree of cost at most C that covers all nodes in R

Certifier: Can easily check in polynomial time that

a- Tree covers on nodes in R (run BFS on the tree)
b- The total cost of the tree is at most C

A and b can be easily done in polynomial time. → Steiner Tree Problem ϵ NP

2- Choose vertex cover for our reduction
3- Will show that Vertex Cover $\leq_p$ Steiner Tree Problem
We will start with an instance of the vertex cover problem (Is there an vertex cover of size at most k in G) and will construct G' such that G has a vertex cover of size at most k iff G' has a Steiner Tree of cost at most m+k.

Construction of G': G' will have the same set of nodes and edges in G plus a number of new nodes and edges. The nodes in G' that exist in G will belong to the set S. We now introduce a new set of nodes in G':

- We will add one node (r_e) per edge e in G' (adding m nodes in this process). All these nodes will belong to the set R
- We will connect each node to the two ends of the corresponding edge in G'
- We will add one more node r_0 in G' and will connect r_0 to all the nodes in the set S in G'
- All edges in G' will have a cost of 1

Proof of correctness for the reduction step:

A- If we have a vertex cover of size k in G, we can produce a Steiner tree of cost m+k in G'. This can be done by using the following edges in the Steiner Tree
  a. k edges that connect r_0 to the node in G' corresponding to those k nodes that form a vertex cover of size k in G.
  b. m edges that connect each of the r_e nodes in G' to the node that covers in e in G.

  This will result in a tree of cost m+k that covers all nodes in the set R.

B- If we have a Steiner tree of cost m+k in G' we can produce a vertex cover of size k in G. This can be done by putting all the nodes in the set S that are part of the Steiner tree in the vertex cover set. (m of the edges will be needed to connect r_e nodes to one of the S nodes. The other k edges in the tree will be connecting these S nodes (that are part of the tree) to other S nodes on the tree (for example through node r_0). Since these S nodes have direct connections to all nodes in the set R, then the nodes corresponding to these S nodes in G will form a vertex cover of size k.)