CSC 373 Lecture 18

- Some simple reductions
- NP sets and NP completeness
- Reducing search/optimization to corresponding decisions problems
- Building a tree of NP complete problems

Some relatively easy transformations

- Vertex cover transforms to independent set and conversely, independent set transforms to vertex cover. Independent set and clique transform to each other.
- Note: these are NP complete problems and all such problems can theoretically be reduced to each other. But here the reduction in both directions is immediate.
- SAT to 3-SAT (Clearly here the converse holds.)
- 3-SAT to IS (independent set). Why noteworthy?

NP Sets (decision problems)

- What do these sets (say SAT and CLIQUE) have in common? They both can be easily "**verified**" by a succinct "**certificate**".
- For example, suppose I am "all powerful" (or perhaps just as good, suppose I am just a very lucky at guessing).
- Then if I want to prove that F is in SAT, I show you a satisfying truth assignment (call it *tau*) and then you (or an efficient algorithm) can easily verify that *F* is satisfied by *tau*. *Tau* is the succinct certificate.
- Similarly if I want to convince you that *(G,k)* is in CLIQUE, then I show you a subset of *k* nodes *V'* and you verify that *V'* is a clique in *G*.

The definition of an NP set

- Let *L* be a set (i.e. a subset of strings over some finite alphabet). Then *L* is in ***NP*** if there exists a polynomial time predicate (i.e. 0-1 valued function) *R(x,y) and polynomial q such that x in L* iff there exists a *y:* $|y| <= q(|x|)$ and *R(x,y)* is true (i.e. $R(x,y) = 1$). That is, every x in L has a succint certificate y (where the poly q defines "succinct") that allows for efficiently verifying that *x* in *L (*where poly time *R* defines efficient verification) .

All the problems studied to date have corresponding NP decision problems

- (Job) Interval scheduling decision problem: For a set *S* of weighted intervals (resp. jobs for the JISP problem), and bound *W*, does there exist a subset of intervals (jobs) with profit at least *S*.
- The knapsack decision problem: For a set of items, size bound *W* and value bound *V*, does there exist a subset of items with total size at most *W* and value at least *V.*
- For sets in polynomial time (i.e. in ***P***) no certificate is needed. Clearly *P* is a subset of *NP*.

NP Complete Sets

- Let <= be a poly time reducibility (or poly time transformation). We will say that a set (decision problem) *L* is ***NP hard*** if for every *L'* in *NP* , *L'* <= *L*. Hence if *L* is *NP* hard but is also in *P*, then *P = NP*
- *L* is ***NP complete*** if *L* is in *NP* and *NP* hard. Hence *P = NP* iff there is any *NP* complete problem that is in *P*.
- Why do we religiously believe that *P* is not equal to *NP*? Because there are thousands of NP complete problems that have been thought about independently before and after the concept was defined and no one has been able to find a polynomial time algorithm for them. Moreover, the best algorithms for these natural *NP* problems are all exponential time (i.e. c^n for some $c > 1$)

The tree of NP completeness

- How do we show that a set *L* is *NP* complete? Usually (but not always) it is relatively easy to show that *L* is in *NP*. Usually it is the NP hardness that can sometime be quite non trivial to show. In fact, one might wonder how we show that any set *L* is *NP* hard since it requires showing something about every *L'* in *NP*. But suppose we do have one set *L* which is *NP* complete. Then if if we find another *L** in *NP* such that $L <= L^*$ then L^* is also NP complete by the transitivity of <=. So starting with some *NP* complete *L* we can start to evolve a tree of *NP* complete problems.