CS 331 NP Notes 1

NP Computational Intractability

- Recall: an algorithm is efficient if it has a polynomial running time.
- Certain problems are extremely hard and cannot be solved by efficient algorithms.
- We do **not** know any polynomial time algorithms for these problems, and we **cannot** prove that no polynomial time-algorithm exists.
- A large class of these problems has been characterized and has been proven to be equivalent in the following sense: a polynomial-time algorithm for any one of them would imply the existence of a polynomial time algorithm for all of them. These problems are known as the NP-Complete problems.

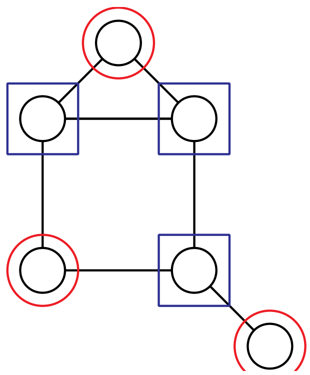
Polynomial-Time Reduction: is the basic technique that we will use to explore the space of computationally hard problems. Using reduction, we can formally express statements like, "problem X is at least as hard as problem Y."

- **Definition of Reduction:** Let X and Y be two problems.
  $Y \leq_P X$ (meaning Y can be reduced to X in polynomial time)
  if and only if an arbitrary instance of problem Y can be solved using a polynomial number of standard computational steps, plus a polynomial number of calls to a *black box* which solves X i.e. how many times you call the black box
  Explanation of the definition: To solve an instance of Y, you can do polynomial amount of work (regular kind of algorithm to create an instance of X) but you are able to call a black box that can solve instances of X.
    - The black box is sometimes called the *oracle* -- is not a realistic model of computation.
        - The oracle submits the question and receives the answer
            - Question must be asked in a "yes" or "no" format
                - What we call a *decision version* of a problem
                    - Instead of returning the complete solution we simply return whether a solution exists or not
- Other notes about $Y \leq_p X$
    - This means Y is polynomially-reducible to X
    - Also means X is at least as hard as Y
    - Also means Y can be solved using a polynomial number of steps plus a computational number of calls to X's black box.

NP-Complete Example: Independent Set

- **Problem Definition:** Given a graph $G = (V, E)$, we say that a set of nodes $S \subseteq V$ is independent if no two nodes in S are joined by an edge, i.e., nodes in S are not adjacent.
- Note that finding small independent sets in a graph is easy but finding **the largest independent set** is hard.
- **Goal:** Find the largest independent set

    - i.e. find the maximum number of nodes such that no two nodes are joined by an edge.
- We need to rephrase our problem as a decision problem so that it can communicate with the oracle



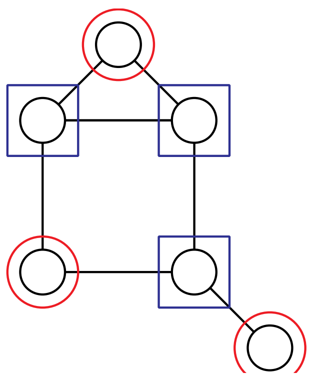
- Independent Set
    - Does G contain an independent set of size **at least k**?
      if k=2, then the answer is Yes.
      if k=4, then the answer is No.

To illustrate the basic strategy for relating hard problems to one another, we consider another fundamental graph problem for which no efficient algorithm is known.

<u>**Vertex Cover**</u>

- **Problem Definition:** Given a graph $G = (V, E)$, we say that a set of nodes $S \subseteq V$ is a Vertex Cover if every edge $e \in E$ has at least one end in S, i.e., S covers all edges.
- Note that finding largest Vertex cover in a graph is easy but finding **the smallest Vertex cover** set is hard.
- **Goal:** Minimize the number of vertices used to cover E.
    - i.e. minimize the number of nodes which can successfully account for every edge in the graph
- Decision version
    - Does G contain a vertex cover of size **at most k**?
      if k=3, the answer is Yes.
      if k=4, the answer is also Yes (we can also consider the node at the bottom left).

Relationship Between Independent Set and Vertex Cover



Independent Set
Vertex Cover

Note: we do not know how to solve either Independent Set (IS) or Vertex Cover (VC) in polynomial time; but what can we say about their relative difficulty?
We will show that they are equivalently hard: $IS \leq_P VC$ and VC$\leq_P IS$.

Lemma 8.1: Suppose $Y \leq_P X$. If X can be solved in polynomial time, then Y can be solved in polynomial time

- This makes sense, since we have a polynomial number of steps in our reduction and a polynomial number of calls to X then it stands to reason that for some P; $P^A \cdot P^B \cdot P^C = P^{A+B+C}$ Where P^A is the time taken for the reduction, P^B is the number of calls to X and P^C is the time taken for X to run IF X runs in polynomial time.

Lemma 8.2: Suppose $Y \leq_P X$. If Y cannot be solved in polynomial time, then X cannot be solved in polynomial time

- Since $Y \leq_P X$ is equivalent to "X is at least as hard as Y" then it must be the case that if Y cannot be solved polynomially then X cannot be solved polynomially either since the two are equally hard. I.e. the "hardness" spreads from Y to X.
  Note that Lemma 8.2 is the contrapositive of Lemma 8.1. [the cotrapositive of $p \rightarrow q$ is $\sim q \rightarrow \sim p$].

Lemma 8.3: Let $G = (V, E)$ be a graph, then S is an independent set **if and only if** $V - S$ is a vertex cover.

Proof:
$\rightarrow$
- First, suppose that S is an independent set.
- Let $e = (u, v)$ be an arbitrary edge in G.
- Since S is independent, it cannot be the case that both u and v are in S as that would contradict the claim that S is an independent set.

- Therefore, one of the endpoints of e must lie in the set $V - S$.
- Therefore, since S is an independent set, it follows that this must be true $\forall_e \in G$. i.e., every edge has at least one end in $V - S$. By definition, $V - S$ is a vertex cover.

$\leftarrow$

- Suppose $V - S$ is a vertex cover.
- Consider any two nodes u and $v \in S$.
- If u and v were joined by an edge, then not both ends of the edge would lie in $V - S$, contradicts our assumption that $V - S$ is a Vertex Cover.
- No two nodes in S are joined by an edge.
- So, S is an independent set.

We can conclude that IS and VC are closely related to each other. ■

Is Independent Set $\leq_P$ Vertex Cover?

- i.e. can independent set be reduced to vertex cover in polynomial time?
  - To find out we need to show:
    - The problems are strongly related
    - Need to apply the definition

Reduction $I.S. \rightarrow V.C.$

According to the graph above, the relationship between V.C. and I.S. is such that

$|V.C.| + |I.S.| = V$

Therefore, $|V.C.| = V - |I.S.|$

To show that VC is reducible in polynomial time to IS, we have to satisfy the definition.

- Come up with an arbitrary instance of VC: a graph G, a target K
- Compute $k' = |V| - k$
- Call the black box for IS
- With input G, k′, the black box returns either Yes or No (it says whether there is an IS in G of size k′ or not).
- If the returned answer is ("yes" or "no") for independent set..

This is the algorithm for solving VC if we have a have (able to find) a black box that can solve instance of IS.

The time of the algorithm is polynomial.

- $k' = |V| - k$ - polynomial time
- call black box - polynomial time, we only called it once.
- receive output (yes or no)- polynomial

<u>8.4:</u> IS$\leq_P$ VC

<u>8.5:</u> VC$\leq_P$ IS