CS 440 More on Reduction, NP and NP-Complete

- Polynomial Reductions

  - Definition: Polynomial Turing Reduction
    - Let A and B be two problems. We say that A is *polynomially Turing reducible* to B, denoted $A \leq_T^P B$, if there exists an algorithm for solving A in a time that would be polynomial if we could solve arbitrary instances of problem B at unit cost.
    - In other words, the algorithm for solving problem A may make whatever use it chooses of an imaginary algorithm that can solve problem B at unit cost.

  - Definition: Polynomial Turing Equivalence
    - A and B are *polynomially Turing equivalent*, denoted $A \equiv_T^P B$, if $A \leq_T^P B$ and $B \leq_T^P A$.

- Example: Hamiltonian cycle problem (HAM) and the Hamiltonian cycle decision problem (HAMD).

  - It turns out that it is not significantly harder to find a Hamiltonian cycle than to decide if a graph is Hamiltonian.

  - Theorem: HAM $\equiv_T^P$ HAMD
    - Proof: (must show two things: 1. HAMD $\leq_T^P$ HAM and 2. HAM $\leq_T^P$ HAMD)

  1. HAMD $\leq_T^P$ HAM

```
function HamD(G: graph)
    σ ← Ham(G)
    if σ defines a Hamiltonian cycle in G then
        return true
    else
        return false
```

2. HAM $\leq_T^P$ HAMD

```
function Ham(<N,A>: graph)
    if HamD(<N,A>) = false then
        return "no solution"
    for each e ∈ A do
        if HamD(<N,A - {e}>) then
            A ← A - {e}
    σ ← sequence of nodes obtained by following the unique cycle remaining
    return σ
```

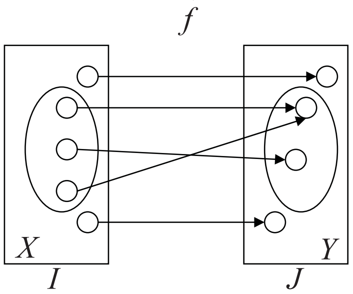
From 1 and 2, HAM $\equiv_T^P$ HAMD.

- Theorem: Consider two problems A and B. If A $\leq_T^P$ B and if B can be solved in polynomial time then A can also be solved in polynomial time.

  - It follows that a polynomial time algorithm exists to find a Hamiltonian cycle if and only if a polynomial time algorithm exists to decide if a graph is Hamiltonian.

  - This is typical of many interesting problems which are polynomially equivalent to a similar decision problem
    ➔ *Decision reducible*: If a problem interests you is not a decision problem, you probably can find a similar decision problem that is polynomail equivanent.

- Polynomial Many-to-One Reductions
  - Definition: Polynomial many-to-one reduction
    - Let X and Y be two decision problems defined on sets of instances I and J, respectively. Problem X is *polynomially many-to-one reducible* to problem Y, denoted $X \leq_m^P Y$, if there exists a reduction function f: I → J computable in polynomial time such that $x \in X$ if and only if $f(x) \in Y$ for any instance $x \in I$ of problem X.
  - Definition: Polynomial many-to-one equivalence
    - When $X \leq_m^P Y$ and $Y \leq_m^P X$, X and Y are *polynomially many-one equivalent*, denoted $X \equiv_m^P Y$
  - In other words, the reduction function maps all yes-instances of problem X onto yes-instances of problem Y and all no-instances of problem X onto no-instances of problem Y.



- Theorem: If X and Y are two decision problems such that $X \leq_m^P Y$ then $X \leq_T^P Y$.
  - Proof: Imagine solutions to Y can be obtained at unit cost by a call on DecideY and let f be the reduction function between X and Y computable in polynomial time. Consider the following algorithm:

```
function DecideX(x)
    y ← f(x)
    if DecideY(y) then
        return true
    else
        return false
```

By definition of the reduction function, this algorithm solves problem X. Because the reduction function is computable in polynomial time, it solves problem X in polynomial time since calls to DecideY can be counted at unit cost.

- Sometimes if we want to prove that $X \leq_T^P Y$, proving $X \leq_m^P Y$ is sufficient.

- Theorem: HAMD $\leq_T^P$ TSPD (Traveling Salesperson Decision problem)
  - Proof:
    - Let $G = \langle N, A \rangle$ be a graph with n nodes. We would like to decide if it is Hamiltonian.
    - Define f(G) as the instance of TSPD consisting of the complete graph $H = \langle N, N \times N \rangle$,
    - Define the cost function for an edge in H: $c(u, v) = \begin{cases} 1, & \text{if } (u, v) \in A \\ 2, & \text{otherwise} \end{cases}$

      and the bound L = n.
    - Any Hamiltonian cycle in G translates into a tour in H that has cost exactly n. On the other hand, if there is no Hamiltonian cycle in G, any tour in H must use at least one edge of cost 2, and thus be of total cost at least n+1.
    - Therefore, G is a yes-instance of HAMD if and only if $f(G) = \langle H, c, L \rangle$ is a yes-instance of TSPD.
    - This proves that HAMD $\leq_m^P$ TSPD ➔ HAMD $\leq_T^P$ TSPD

- NP-Complete Problems
  - Definition: A decision problem X is NP-complete if:

    $X \in NP$; and

    $Y \leq_T^P X$ for every problem $Y \in NP$ (i.e., X is NP-Hard)

- Theorem: Let X be an NP-complete problem. Consider a decision problem $Z \in NP$ such that $X \leq_T^P Z$. Then Z is also NP-complete.

  - Proof: Z must meet the two conditions of the definition of NP-completeness.
    - $Z \in NP$. It is in the statement of the theorem.
    - Z is NP-Hard (i.e., $Y \leq_T^P Z$ for every problem $Y \in NP$)

      Consider an arbitrary $Y \in NP$. Since X is NP-complete and $Y \in NP$, it follows that $Y \leq_T^P X$.

      We know by the statement of the theorem that $X \leq_T^P Z$.

      By *transitivity* of polynomial reductions it follows $Y \leq_T^P Z$.

- To prove that Z is NP-complete,
  - We prove that Z ∈ NP by verifying a given solution to Z in polynomial time if it is not in the statement of the theorem.
  - We choose an *appropriate* problem from the pool of several thousand problems already known to be NP-complete and show that it is polynomially reducible to Z.

- The previous statement works nicely once the process is underway (i.e., once at least one NP-complete problem has been identified)
  - Q: How do we find the *FIRST* NP-complete problem? (What is it?)
  - Good news: We don't have to find it. But we have to know what it is.

- Satisfiability problem (SAT)
  - Definition: A Boolean formula is *satisfiable* if there exists at least one way of assigning values to its variables so as to make it true.
  - Example: $(\neg(p \vee q)) \vee (p \wedge q)$ YES $\neg p \wedge (p \vee q) \wedge \neg q$ NO
  - SAT ∈ NP

- SAT-CNF
  - A literal is either a Boolean variable or its negation.
  - A clause is a literal or a disjunction of literals.
  - A Boolean formula is in *conjunctive normal form* (CNF) if it is a clause or a conjunction of clauses.
  - It is in k-CNF for some positive integer k if it is composed of clauses, each of which contains at most k literals.
    - Examples:

      $(p \vee \neg q \vee r) \wedge (\neg p \vee q \vee r) \wedge q \wedge \neg r$ CNF? k = ?

      $(p \vee q \wedge r) \wedge (\neg p \vee q \wedge (q \vee r))$ CNF? k = ?
  - SAT-CNF is a restriction of SAT to Boolean formulas in CNF.
  - For any positive k, SAT-k-CNF is the restriction of SAT-CNF to Boolean formulas in k-CNF.

- Theorem (due to Stephen Cook, 1971):
  - SAT-CNF is NP-complete.

- NP-Completeness Proofs
  - Theorem: SAT is NP-complete
    - Proof:
      1. SAT $\in$ NP.
         Given an assignment of variables, we can verify the Boolean formula is true or false in polynomial time.
      2. (must show SAT-CNF $\le_T^P$ SAT)
         Boolean formulas in CNF are special cases of general Boolean formulas
         ➔ SAT-CNF $\le_T^P$ SAT
  - Theorem: SAT-3-CNF is NP-complete
    - Proof:
      1. SAT-3-CNF $\in$ NP.
         Given an assignment of variables, we can verify the Boolean formula is true or false in polynomial time.
      2. (must show SAT-CNF $\le_T^P$ SAT-3-CNF)
         Proof will be given in the class if time allows.

- More basic known NP-complete problems for NP-completeness proofs
  - Basic known NP-complete problems that can be used to show some other problem is NP-complete

```
            SAT-CNF
              ↓                  ↓: ≤T^P
           SAT-3-CNF
           ↙       ↘
        3DM          VC
         ↓          ↙  ↘
   PARTITION    HAM  CLIQUE
```

  - VC (Vertex Cover): Given a graph and an integer K, is there a set of less than K vertices that touches all the edges?
    Instance: A graph $G = (V, E)$ and a positive integer $K \le |V|$
    Question: Is there a vertex cover of size K or less for G?
    (i.e., Is there a subset $V' \subseteq V$ such that $|V'| \le K$ and, for every edge $\{u, v\} \in E$, at least one of u and v belongs to V'?)

- CLIQUE
  Instance: A graph $G = (V, E)$ and a positive integer $J \leq |V|$
  Question: Does G contain a clique of size of J or more?
  (i.e., Is there a complete subgraph of G that has at least J vertices?)

- 3DM: 3-Dimensional Matching
  Instance: A set $M \subseteq W \times X \times Y$, where W, X, and Y are disjoint sets and $|W| = |X| = |Y| = q$
  Question: Does M contain a matching?
  (i.e., Is there a subset $M' \subseteq M$ such that $|M'| = q$ and no two elements of M' agree in any coordinate.)
  Note that 2DM is not NP-complete.

- PARTITION: Given a set of integers, can they be divided into two sets whose sum is equal?
  Instance: A finite set A and a *size* $s(a) \in Z^+$, for all $a \in A$
  Question: Is there a subset $A' \subseteq A$ such that $\Sigma_{a \in A'} s(a) = \Sigma_{a \in A-A'} s(a)$?

- Example

  - Knapsack problem
    Instance: A finite set U,
    a weight $w(u) \in Z^+$ and a value $v(u) \in Z^+$, for all $u \in U$,
    a capacity $W \in Z^+$, and a value goal $K \in Z^+$.
    Question: Is there a subset $U' \subseteq U$ such that
    $\Sigma_{u \in U'} w(v) \leq W$ and $\Sigma_{u \in U'} v(u) \geq K$?

  - Theorem: Knapsack problem is NP-complete
    - Proof:
      1. Show Knapsack problem $\in$ NP.
      2. Reduce a well-known NP-complete problem to Knapsack problem. (But, which one?)

- NP-Hard Problems
  - A problem X is *NP-hard* if there is an NP-complete problem Y that can be polynomially Turing reduced to it (i.e., $Y \leq_T^P X$)
  - Note that any polynomial-time algorithm for X would translate into one for Y. Since Y is NP-complete, this would imply that P = NP.
    - Under the assumption of P ≠ NP, no NP-hard problem can be solved in polynomial time.
  - Note that NP-hard contains non-decision problems.
    - Example:
      - TSP
      - 0-1 knapsack problem, time $\in \Theta(nW)$, is it polynomial? What if $W \in \Theta(2^n)$?
  - Further note that there are decision problems that are NP-hard but are believed to not be in NP and thus not in NP-complete.
    - Example:
      - COLE: exact coloring, given a graph G and an integer k, can G be painted with k colors but no less?