## Introduction to NP-completeness

- The classes of P and NP
- Polynomial reduction
- NP-complete problems and proofs
- NP-hard problems
- Approximate algorithms

## Class P problems

- Practical considerations
- What kind of problems can be solved practically or efficiently
  - An algorithm is *efficient* if there exists a polynomial $p(n)$ such that the algorithm can solve any instance of size n in a time in $O(p(n))$
- Decision problems
  - For those problems, the answer is either yes or no
- $P$ is the class of decision problems that can be solved by a polynomial-time algorithm

## Class NP problems

- We are talking about *polynomially verifiable* properties
- Example
  - Problem: give a graph, decide if it is a Hamiltonian
    - An undirected graph is a Hamiltonian if it contains a Hamilton cycle: a path starts with some node, visits each node exactly once, and returns the starting node
  - Verification
    - Given a path, we can efficiently verify if it is a Hamilton cycle

## Definition of class NP

- $NP$ is the class of decision problems $X$ that admit a proof system $F \subseteq X \times Q$ such that there exists a polynomial $p(n)$ and a polynomial-time algorithm $A$ such that
  - For all $x \in X$, there exists a $q \in Q$ such that $<x, q> \in F$ and moreover the size of $q$ is at most $p(n)$, where $n$ is the size of of $x$
  - For all pairs $<x, q>$, algorithm $A$ can verify whether or not $<x, q> \in F$. In other words $F \in P$.

## Examples of class NP problems

- Is a graph G Hamiltonian?
  - X is the set of all Hamiltonian graphs
  - Q is set of sequence of graph nodes
  - Define $<G, \sigma> \in F$ if and only if nodes $\sigma$ specifies a Hamiltonian cycle in Graph G
- Is a number n a composite number?
  - X is the set of all composite numbers
  - Q = N is the proof space
  - $F = \{ <n, q> \mid 1<q<n$ and q divides $n \}$

## P and NP

- Theorem $P \subseteq NP$
  - Consider an arbitrary decision problem $X \in P$. Let $Q = \{0\}$ and $F = \{<x, 0> \mid x \in X\}$
    - For any $x \in X$, q is 0
    - For any $<x, q>$, we can directly verify it by verifying if $x \in X$ and q=0

## Polynomial Reduction

- Let A and B be two problems. We say 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.
- When $A \leq_T^p B$ and $B \leq_T^p A$ both hold, we say that A and B are polynomially Turing equivalent and write $A \equiv_T^p B$

$HAM \equiv_T^p HAMD$

- HAM find a Hamilton cycle in a graph
- HAMD decides if a graph is Hamiltonian

```
HamD(Graph G)
{
  c = Ham(G);
  if (c is a Hamiltonian cycle in G)
    return true;
  else
    return false;
}
```

$HAMD \leq_T^p HAM$

```
Ham(Graph G=<N, A>)
{
  if (!HamD(G))
    return no solution;

  for each edge e in A
     if (HamD(N, A-{e}))
        A = A –{e};
  return the unique cycle
  remaining in G
}
```

$HAM \leq_T^p HAMD$

## Polynomial many-one reduction

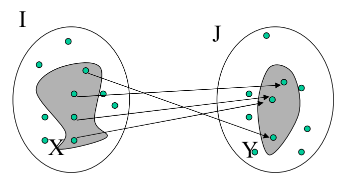
- Let $X$ and $Y$ be decision problems defined on sets of instances $I$ and $J$. Problem $X$ is polynomially many-one reducible to problem $Y$ if there exists a function $f: I \rightarrow J$ computable in polynomial time such that $x \in X$ if and only if $f(x) \in Y$ for any instance $x \in I$ of problem $X$. This is denoted $X \leq_m^p Y$

  and function f is called the reduction function.
- When $X \leq_m^p Y$ and $Y \leq_m^p X$ both hold, we say that $X$ and $Y$ are polynomially many-one equivalent and we write as $X \equiv_m^p Y$

## Theorem

- If X and Y are two decision problems and such that $X \leq_m^p Y$, then $X \leq_T^p Y$

```
DecideX(x)
{
  y = f(x);
  return DecideY(y);
}
```




## TSP and TSPD

- TSP
  - Given a graph with weighted edges, find a tour that begins and ends at the same node after having visited each node exactly once and whose the total cost of tour is the minimum possible; The answer is undefined is no such tour exists
- TSPD
  - Decide whether or not there exists a valid tour whose total cost does not exceed L.

$HAMD \leq_m^p TSPD$

- Proof
  - Let G = <N, A> be a graph with n nodes. We'd like to decide if it is Hamiltonian. Define f(G) as an instance of TSPD consisting of the complete graph H=<N, N×N>. The cost function is as follows

$$c(u,v) = \begin{cases} 1 & \text{if } \{u,v\} \in A \\ 2 & \text{otherwise} \end{cases}$$

  - Let the bound L be n.
  - Any Hamiltonian cycle in G translates into a tour in H that has exactly cost n.
  - If there is no Hamiltonian cycles in G, any valid tour in H must use at least one edge of cost 2 and the total cost will exceed L. Therefore, G is a yes instance of HAMD iff H is a yes instance of TSPD.

## NP-complete problems

- A decision problem $X$ is *NP-complete* if
  - $X \in NP$ and
  - $Y \leq_T^p X$ for every problem $Y \in NP$
- 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
- We don't know if P=NP but we conjecture that P≠NP

## SAT-CNF is NP-complete

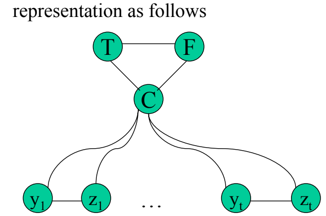
- SAT
  - Given a Boolean formula, decide whether or not it is satisfiable
- CNF
  - A literal is either a Boolean variable or its negation
  - A clause is a literal or disjunction of literals
  - A CNF formula is either a clause or conjunction of clauses
  - A k-CNF formula is a CNF formula with clause contains at most k literals
- Cook's Theorem: SAT-CNF is NP-complete

## SAT-3-CNF is NP-complete

- First SAT-3-CNF ∈ NP
- Second, $SAT-CNF \leq_T^p SAT-3-CNF$
  - For any Boolean formula β ∈ CNF, we construct efficiently a Boolean formula γ = f(β) ∈ 3-CNF that is satisfiable is and only if β is satisfiable
    - Transform each clause $x$ in β to $y$ in γ as follows, assuming that the clause contains k literals
      - If k<=3, directly map: $y=x$
      - If k=4. Let $x = l_1 + l_2 + l_3 + l_4$ and u be a new Boolean variable
        - » Take $y = (l_1 + l_2 + u)(\overline{u} + l_3 + l_4)$
      - If k>=4, let $x = l_1 + l_2 + ... + l_k$ and $u_1, u_2, ... u_{k-3}$ be new Boolean variables
        - » Take $y = (l_1 + l_2 + u_1)(\overline{u}_1 + l_3 + u_2)...(\overline{u}_{k-3} + l_{k-1} + l_k)$
    - We can show that given any fixed values of the literals in $x$, $x$ is true if and only if $y$ is satisfiable with a suitable assignment for the $u_i$'s

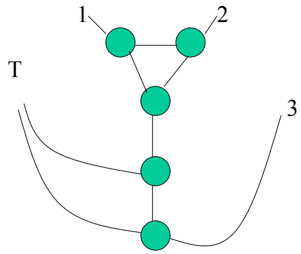
## 3COL is NP-complete

- 3COL: given a graph G, is G 3 colorable?
- First 3COL ∈ NP
- Second, $SAT-3-CNF \leq_T^p 3COL$
  - Given a 3CNF formula γ, create a graph as follows
    - For all Boolean variables $x_1, x_2, .., x_t$, create a graph representation as follows



## Widget for clause

- For each clause create a widget and connect to the "Boolean" graph




## NP-hard problems

- A problem is NP-hard if there exists a NP-complete problem Y that can be polynomially Turing reduced to it: $Y \leq_T^p X$

## Approximating algorithms

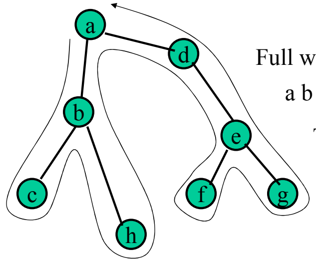
- It's hard to find a practical algorithm to solve NP-hard problems
- Sometimes we are satisfied approximate solutions
  - The solution may be within a range of the optimal, may be not

## The metric traveling salesperson

- A special case of TSP which satisfies metric property.
  - A distance matrix is said to have metric property if the triangle inequality holds: for any three towns i, j, and k
    - distance(i, j) <= distance(i,k)+distance(k,j)
- An approximate algorithm
  1. Find a minimum spanning tree
  2. Build a tour through preorder search starting and ending at the root
- The algorithm find a tour of cost <= 2*minimum possible cost

## Proof

- Let $H^*$ denote an optimal tour and H is the tour returned by the approximation algorithm. Then $c(T) \le c(H^*)$. We want $c(H) \le 2c(H^*)$.
- A full walk W of T lists the vertices when they are first visited and also whenever they are returned after visit a subtree. We have c(W)=2c(T)
- The tour can be generated from the walk W by deleting repeating nodes, which does not increase the cost, i.e., c(H) <= c(W)




Full walk

a b c b h b a d e f e g e d a

The preorder

a b c h d e f g a