NP-Complete Problems

Very very short version

(Please go over the lesson)

Class *P and class NP*

Loosely speaking:

A problem belong to class P if it can be solved in polynomial time.

A problem belong to class NP if it can be solved in polynomial time using a non-deterministic algorithm.

Which is same as saying

A problem belong to class NP if it can be verified in polynomial time.

Solving vs Verifying

Solve the equation:

$7x^2 - 12x - 352 = 0$

Verify $x = 8$ is a solution to the equation

$7x^2 - 12x - 352 = 0$

Verification takes less time!

Example : A Non-deterministic Algorithm

```
IsPresent(A, x)
// A is an array of items. x is an item.
// Will return true if x is present and false otherwise.
        i <- guess(A, x)   // guess will return the correct index
                           // if x is present in the array A.
        if (A[i] == x)
                return true
        else
                return false
```

$P \subseteq NP$

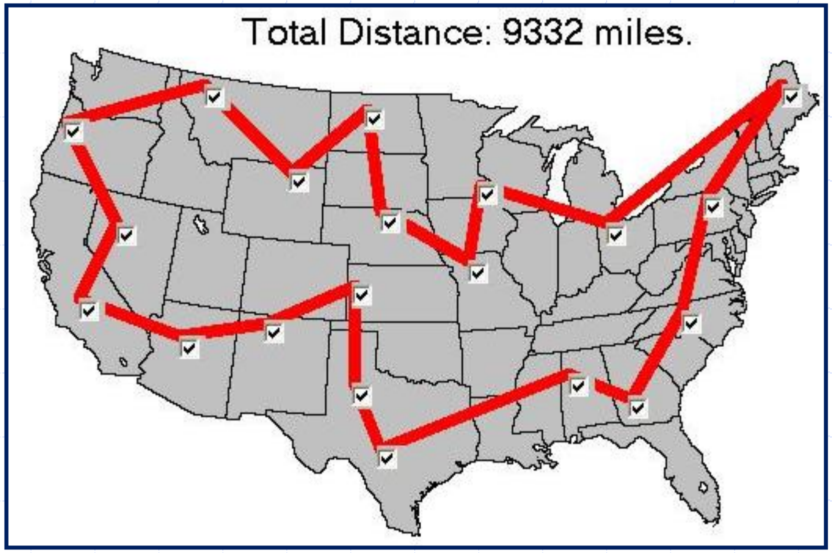
- *Is $P = NP$ we do not know.*

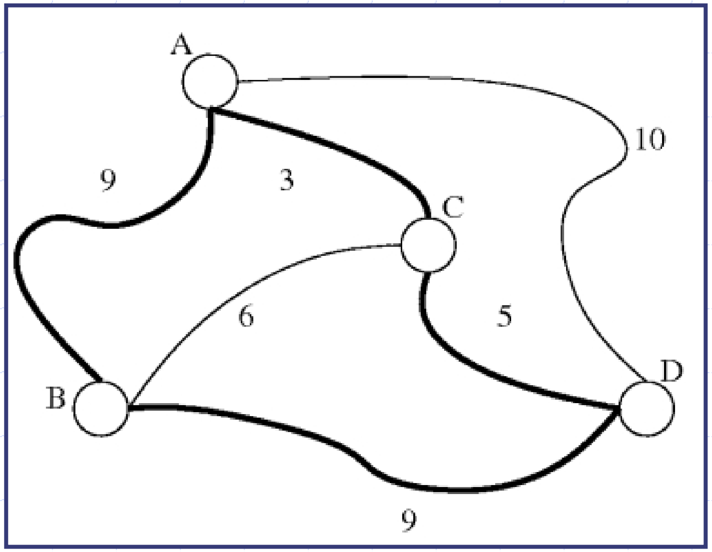
Hamiltonian Cycle And Vertex Covers

- A Hamiltonian cycle in a graph G is a simple cycle that contains every vertex of G. A graph is a Hamiltonian graph if it contains a Hamiltonian cycle.
- Examples. The Herschel graph is not a Hamiltonian graph.
- If $G = (V,E)$ is a graph, a vertex cover for G is a set $C \subseteq V$ such that for every $e \in E$, at least one end of e lies in C.
- Fact. The known algorithms for determining whether a graph is Hamiltonian, and for computing the smallest size of a vertex cover, run in exponential time.

Traveling Salesperson Problem

- *Traveling Saleperson Problem* (TSP): Given a complete graph G with cost function c: E $\rightarrow$ N and a positive integer k, is there a Hamiltonian cycle C in G so that the sum of the costs of the edges in C is at most k? Solution data: a subset of E.





Problems in NP

- HC
- VC
- TSP
- Are in NP

Example: Not in ***NP***

- *PowerSet problem.* Given a set X of size n, a kind of optimization problem concerning the power set of X is to generate all subsets of X ("find the largest possible collection of subsets of X without duplicates"). Whatever method is used, just writing out the output requires at least 2^n steps. A corresponding decision problem is: Given a set X and a collection P, is P = P(X)? Any algorithm that solves this problem must check every element of P to see if it is a subset of X, so the algorithm requires at least 2^n steps in the worst case. So this problem does not belong to P.

  Moreover, verifying correctness requires checking that each set that is output is a subset of X, and this has to be done 2^n times, so it doesn't belong to NP either.

- *Finite Halting Problem and N x N Chess.* We discusssed NxN chess in Lesson 1. Finite Halting Problem: Given a program P and positive integers n, k, does P, when running with input n, produce an output after excecuting k or fewer steps? Neither of these problems belong to P, but it is not known whether they belong to NP.

Reducibility (informal 1)

Let Q denote the problem of finding the area of a square.

Let R denote the problem of finding the area of rectangle.

Given an instance I_Q of Q, we can transform into an instance I_R of R.

I_Q has a solution iff I_R has a solution

- We write $Q \xrightarrow{\text{poly}} R$
- Note that R is "harder" than Q

```
Algorithm areaSquare(double side)
    return (areaRectangle(side, side)
```

Reducibility (informal 2)

Let Q denote the problem computing the distance between two points in 2D.

Let R denote the problem computing the distance between two points in 3D.

Given an instance I_Q of Q, we can transform into an instance I_R of R.

I_Q has a solution iff I_R has a solution

- We write $Q \xrightarrow{\text{poly}} R$
- Note that R is "harder" than Q

```
Algorithm distance2D((x1, y1), (x2, y2))
    return distance3D((x1, y1, 0), (x2, y2, 0))
```

HamiltonianCycle $\xrightarrow{\text{poly}}$ TSP

We show HamiltonianCycle is reducible to TSP

- Given a graph G = (V,E) on n vertices (input for HamiltonianCycle) – notice G is a subgraph of K_n. Obtain an instance H, c, k of TSP as follows: Let H be the complete graph on n vertices (i.e. H is K_n), obtained by adding the missing edges to G. Let
  $c(e) = 0$ if $e \in E$, else $c(e) = 1$. Let $k = 0$.
- Need to show: G has a Hamiltonian cycle if and only if H, c, k has a Hamiltonian cycle with edge cost $\leq k$
- If G has Hamiltonian cycle C, C is Hamiltonian in H also. Since each edge e of C is in G, $c(e) = 0$. So cost sum $\leq k$. Converse: A solution C for H,c,k implies all edges of C have weight 0; therefore, every edge of C also is an edge in G. Therefore C is an HC in G.

HamiltonianCycle $\xrightarrow{\text{poly}}$ TSP

Note: From a practical point of view, all we have to do is to change all non-diagonal values of 1's and 0's in the adjacency matrix of an instance of HC to 0's and 1's to obtain an instance of TSP.

Algorithm HCInstanceToTSPInstance(H)
Input : H an instance of HC as an adjacency matrix.
Output : T an instance of TSP as an adjacency matrix.

```
for i = 0 to n – 1 do
    for j = i + 1 to n – 1 do
        T[i, j ] <- (H[i, j] + 1) % 2
        T[j, i ] <- (H[i, j] + 1) % 2
//Arrays starts with index 0. T initialized with 0 during creation.
//Time complexity O(n^2).
```

HamiltonianCycle $\xrightarrow{\text{poly}}$ TSP

Algorithm isHC(H)
Input : H an instance of HC as an adjacency matrix.
Output : **true** if H is Hamiltonian. **false** otherwise.

T <- HCInstanceToTSPInstance(H)
return isTSP(T, 0)

//isTSP(T, k)
//T an instance of TSP (adjacency matix)
//k a non-negative integer.
//isTSP(T, k) returns **true** if TSP can visit all cities at the cost of k and come back to home city. **false** otherwise.

NP-hard Problems

- A problem Q is ***NP**-hard* if for *every* problem R in ***NP***, R is polynomial reducible to Q.

$$R \xrightarrow{\text{poly}} Q$$

- You can think of them as problems harder than all problems in NP.

NP-Complete Problems

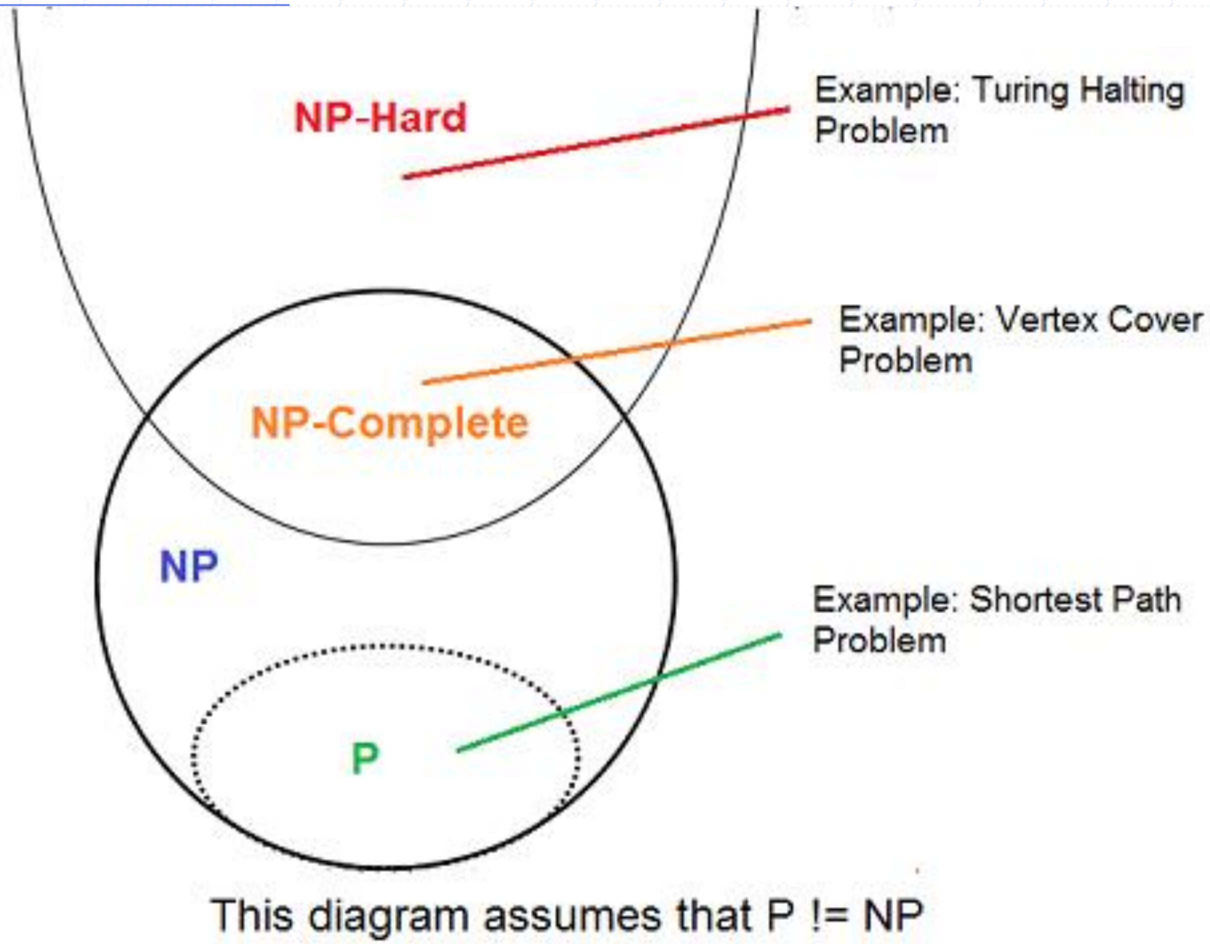
A problem Q is ***NP**-complete* if

Q belongs to *NP*,

and

Q is NP-hard.

NP-Complete Problems



HamiltonianCycle is NP-Complete

This is an outline of a proof that HamiltonianCycle is NP-Complete under the assumption that VertexCover is NP-complete:

- Show HC is in NP.
- Pick VC as the known NP-complete Problem
- Show VC is polynomial reducible to HC (see Slide 20)

Summary: To show Y is NP-Complete

- Show Y is in NP.
- Pick X. A known NP-complete Problem
- Show X is polynomial reducible to Y