NP-Complete Problems

- **Polynomial time** vs **exponential time**
  - –Polynomial $O(n^k)$,
    - –where n is the input size
      - –e.g., number of nodes in a graph, the length of strings , etc
    - –k is a constant
      - –e.g., k=2 in LCS, k=1 in KMP, etc.
  - –Exponential time: 2^n or n^n

–n=	2	10	20,	30
– 2^n	4	1024	1 million	1000 million

–If a computer solves a problem of size n in one hour, now you have a computer 1,000,000 faster, what size of the problem can you solve in one hour?

- –n+20 ($2^{n+20} \approx 1{,}000{,}000 \, 2^n$)
- –The improvement is small.
- –Hardware improves little on problems of exponential running time
- –Exponential running time is considered as "not efficient".

Story

- All algorithms we have studied so far are polynomial time algorithms (unless specified).
- **Facts**: people have not yet found any polynomial time algorithms for some famous problems, (e.g., Hamilton Circuit, longest simple path, Steiner trees).
- **Question**: Do there exist polynomial time algorithms for those famous problems?
- **Answer:** No body knows.

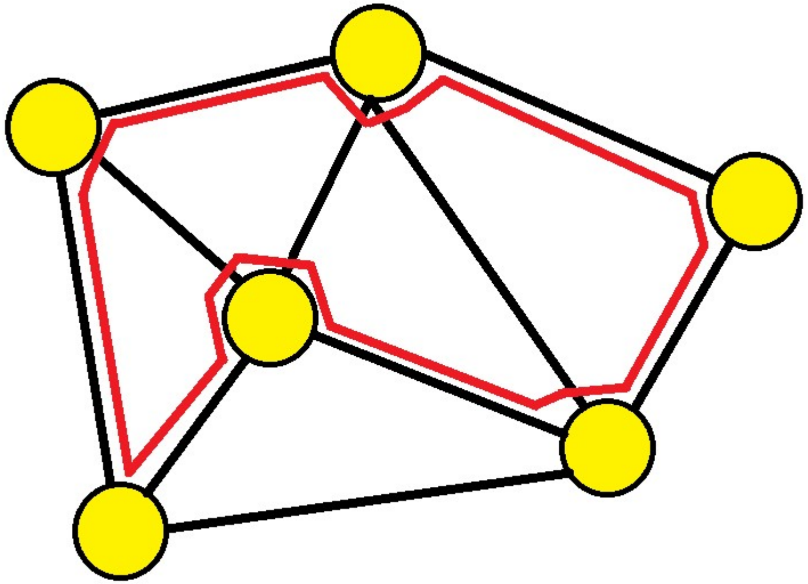
Story

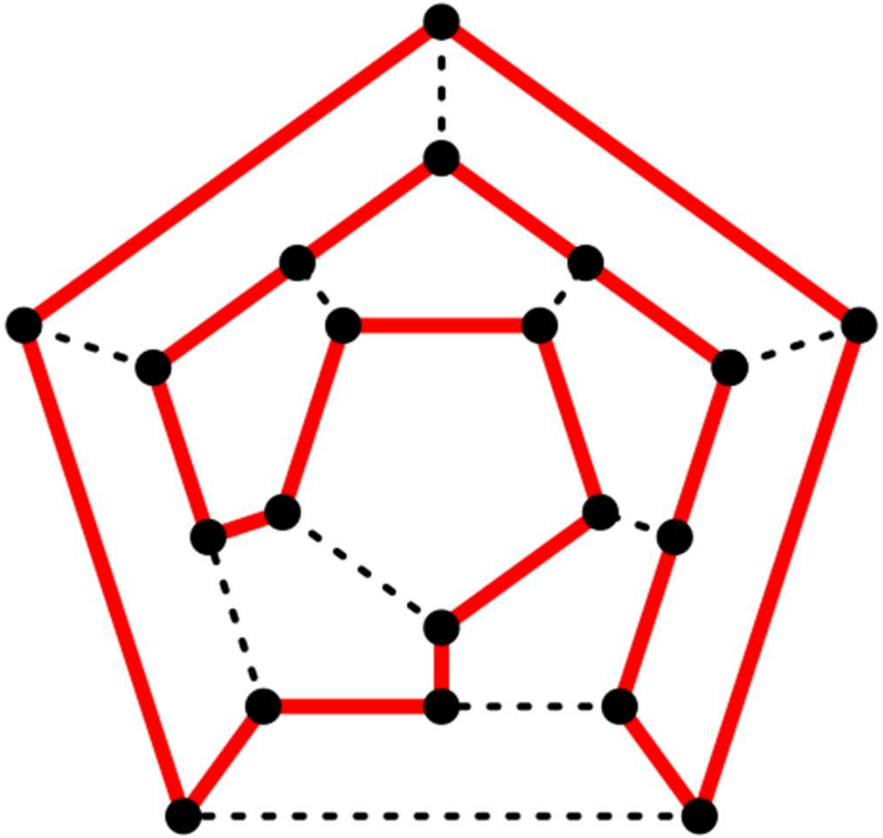
- **Research topic:** Prove that polynomial time algorithms do not exist for those famous problems, e.g., Hamilton circuit problem.
    - *Turing* award
    - Vinay Deolalikar attempted 2010, but failed.
- To answer the question, people define two classes
    - ***P class***
    - ***NP class***.
- To answer if P≠NP, a rich area, NP-completeness theory is developed.

	3			7		5		
		8		5	2			4
6							9	
	2							
1	9			8			2	3
							7	
	8							7
4			8	3		6		
		5		4			8	

	3			7		5		
		8		5	2			4
6							9	
	2							
1	9			8			2	3
							7	
	8							7
4			8	3		6		
		5		4			8	

2	3	1	9	7	4	5	6	8
9	7	8	6	5	2	1	3	4
6	5	4	3	1	8	7	9	2
5	2	7	4	9	3	8	1	6
1	9	6	7	8	5	4	2	3
8	4	3	1	2	6	9	7	5
3	8	9	5	6	1	2	4	7
4	1	2	8	3	7	6	5	9
7	6	5	2	4	9	3	8	1





Class P and Class NP

- Class P contains problems which are solvable in polynomial time.
  - –The problems have algorithms in $O(n^k)$ time, where n is the input size and k is a constant.
- Class NP consists of those problem that are *verifiable* in polynomial time.
  - we can verify that the solution is correct in time polynomial in the input size to the problem.
  - algorithms produce an answer by a series of "correct guesses"
- Example: Hamilton Circuit: given an order of the n distinct vertices $(v_1, v_2, \ldots, v_n)$, we can test if (v_i, v_{i+1}) is an edge in G for $i=1, 2, \ldots, n-1$ and (v_n, v_1) is an edge in G in time $O(n)$ (polynomial in the input size).

Class P and Class NP

- P$\subseteq$NP
  - by definition,
- If we can design a polynomial time algorithm for problem *A*, then *A* is in *P*.
- However, if we have not been able to design a polynomial time algorithm for *A*, then two possibilities:

1. No polynomial time algorithm for *A* **or**
2. We are not smart.

Open problem: P$\neq$NP?

Polynomial-Time Reductions

Suppose a black box (an algorithm) can solve instances of problem X. If we give an instance of X as input the black box will return the correct answer in a single step.

Question: Can we "*transform*" problem Y into X. Can an arbitrary instance of problem Y be solved by the black box?

- We can use the black box polynomial number of times.
- We can use polynomial number of standard computational steps
- If yes, then Y is polynomial-time reducible to X.

$$Y \leq_p X$$

NP-Complete

- A problem X is NP-complete if
  - it is in NP, and
  - any problem Y in NP has a polynomial time reduction to X.
  - it is the hardest problem in NP.
  - If ONE NP-complete problem can be solved in polynomial time, then any problem in class NP can be solved in polynomial time.
- The first NPC problem is *Satisfiability* problem
  - Proved by Cook in 1971 and obtains the Turing Award for this work

Boolean formula

- A boolean formula $f(x_1, x_2, \ldots x_n)$, where x_i are boolean variables (either 0 or 1), contains boolean variables and boolean operations AND, OR and NOT .
- **Clause:** variables and their negations are connected with OR operation, e.g., (x_1 OR NOTx_2 OR x_5)
- **Conjunctive normal form of boolean formula:**

  contains *m* clauses connected with AND operation.

Example:

(x_1 OR NOT x_2) AND (x_1 OR NOT x_3 OR x_6) AND (x_2 OR x_6) AND (NOT x_3 OR x_5).

–Here we have four clauses.

Satisfiability problem

- **Input:** conjunctive normal form with n variables, x_1, x_2, …, x_n.
- **Problem:** find an assignment of x_1, x_2, …, x_n (setting each x_i to be 0 or 1) such that the formula is true (satisfied).
- **Example:** conjunctive normal form is

$$(x_1 \text{ OR NOT} x_2) \text{ AND } (\text{NOT } x_1 \text{ OR } x_3).$$

- The formula is true for *assignment*

$$x_1=1,\ x_2=0,\ x_3=1.$$

Note: for n Boolean variables, there are 2^n assignments.

•Testing if formula=1 can be done in polynomial time for any given assignment.

•Given an assignment that satisfies formula=1 is hard.

The First NP-complete Problem

- Theorem: Satisfiability problem is NP-complete.
  - It is the first NP-complete problem.
  - S. A. Cook in 1971 http://en.wikipedia.org/wiki/Stephen_Cook
  - Won Turing prize for his work.
- Significance:
  - If Satisfiability problem is in P, then ALL problems in class NP are in P.
  - To solve P≠NP, you should work on NPC problems such as satisfiability problem.
  - We can use the first NPC problem, Satisfiability problem, to show other NP-complete problems.

How to show that a problem is NPC?

•To show that problem A is NP-complete, we can

–First, find a NP-complete problem B.

–Then, show that

–IF problem A is P, then B is in P.

–that is, to give a polynomial time reduction from B to A.

***Remarks**: Since a NPC problem,* problem B*, is the hardest in class NP, problem A is also the hardest*

Hamilton circuit and Longest Simple Path

- **Hamilton circuit** : a circuit uses every vertex of the graph exactly once except for the last vertex, which duplicates the first vertex.
  - It was shown to be NP-complete.
- **Longest Simple Path:**
  - Input: $V=\{v_1, v_2, \ldots, v_n\}$ be a set of nodes in a graph G and $d(v_i, v_j)$ the distance between v_i and v_j,
  - Output: a longest simple path from u to v .

- **Theorem: The longest simple path problem is NP-complete.**

Theorem: The longest simple path (LSP) problem is NP-complete.

Proof:

Hamilton Circuit Problem (HC): Given a graph G=(V, E), find a Hamilton Circuit.

We want to show that if the longest simple path problem is in P, then the Hamilton circuit problem is in P.

Design a polynomial time algorithm to solve HC by using an algorithm for LSP.

Step 0: Set the length of each edge in G to be 1

Step 1: for each edge $(u, v) \in E$ **do**

find the longest simple path P from u to v in G.

Step 2: **if** the length of P is n-1 **then** by adding edge (u, v) we obtain a Hamilton circuit in G.

Step 3: **if** no Hamilton circuit is found for every (u, v) **then**

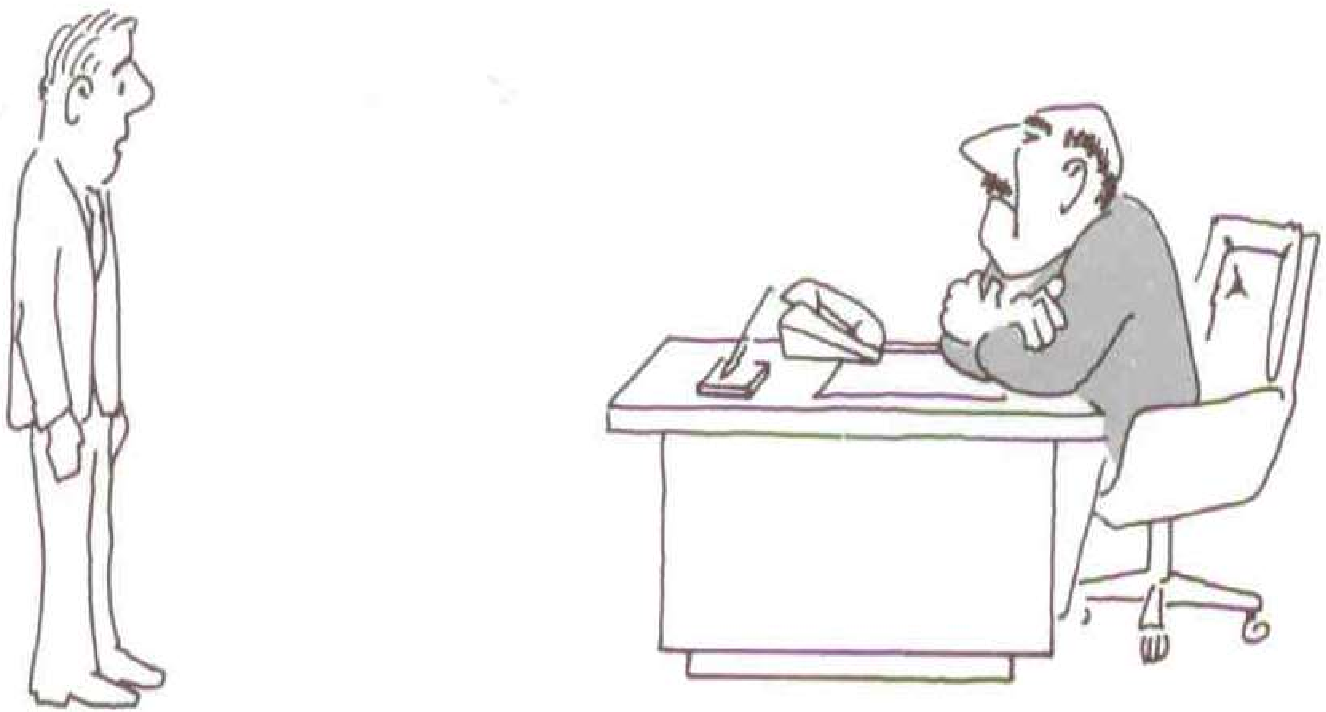
print "no Hamilton circuit exists"

Conclusion:

- if LSP is in P, then HC is also in P.
- Since HC was proved to be NP-complete, LSP is also NP-complete.

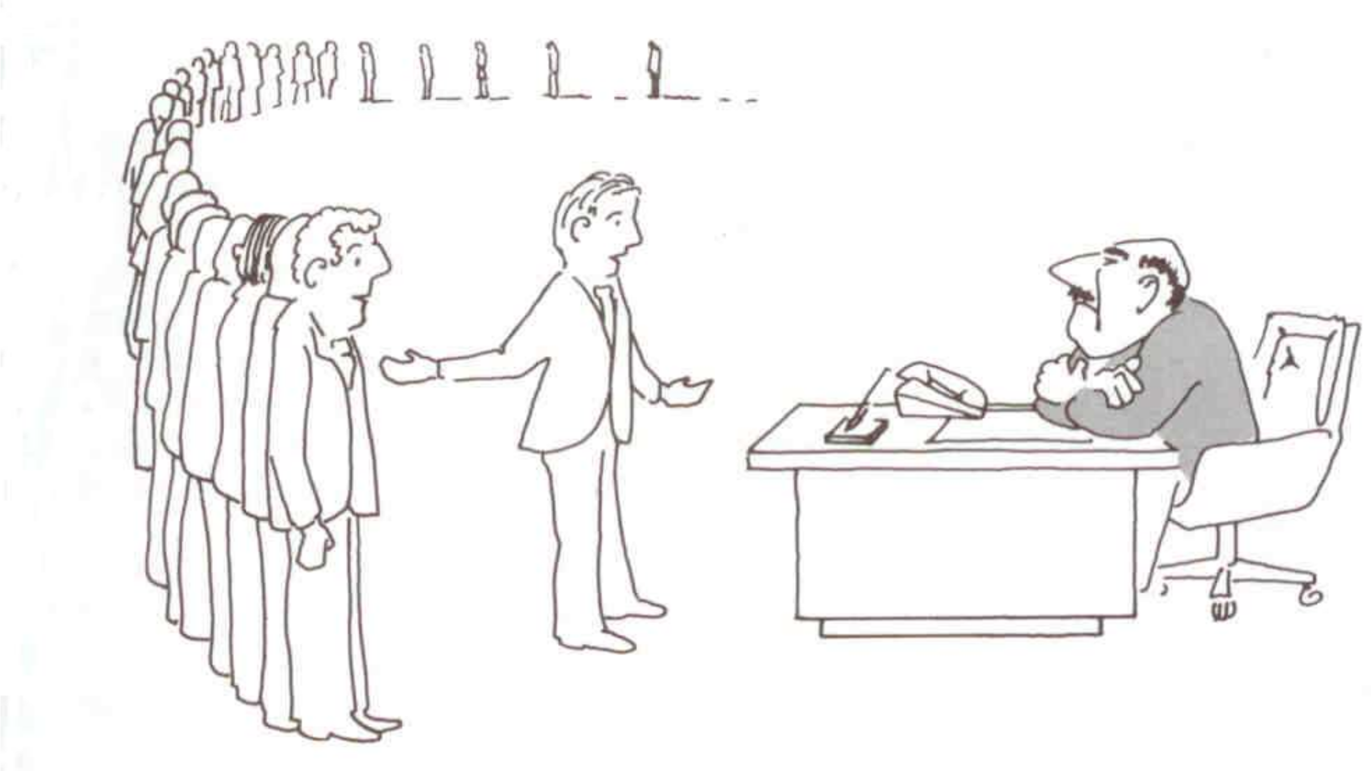
Some basic NP-complete problems

- **3-Satisfiability :** Each clause contains at most three variavles or their negations.
- **Vertex Cover:** Given a graph G=(V, E), find a subset V’ of V such that for each edge (u, v) in E, at least one of u and v is in V’ and the size of V’ is minimized.
- **Hamilton Circuit:** (definition was given before)
- History: Satisfiability→3-Satisfiability→vertex cover→Hamilton circuit.
- Those proofs are very hard.



"I can't find an efficient algorithm, I guess I'm just too dumb."

"I can't find an efficient algorithm, because no such algorithm is possible!"



"I can't find an efficient algorithm, but neither can all these famous people."