P and NP

Inge Li Gørtz

Thank you to Kevin Wayne, Philip Bille and Paul Fischer for inspiration to slides

Overview

- Problem classification
  - Tractable
  - Intractable
- Reductions
  - Tools for classifying problems according to relative hardness

Warm Up: Super Hard Problems

- Undecidable. No algorithm possible.
- Example. Halt (P, x) = true iff and only if P halts on input x.
- Claim. There is no general algorithm to solve Halt(P, x)
- Proof (by contradiction)
  - Suppose algorithm for Halt(P, x) exists.
  - Consider algorithm A(P) which loops infinitely if Halt(P,P) and otherwise halts.
  - Since Halt(P,x) exists for all algorithms P we can use it on A(A) and the following happens:
    - If Halt(A,A) then we loop infinitely.
    - Else (not Halt(A,A)) we halt.

Problem Classification

- Q. Which problems will we be able to solve in practice?
- A. Those with polynomial-time algorithms. (working definition) [von Neumann 1953, Godel 1956, Cobham 1964, Edmonds 1965, Rabin 1966]

Yes	No
Shortest path	Longest path
Min cut	Max cut
Soccer championship (2-point rule)	Soccer championship (3-point rule)
Primality testing	Factoring

Problem Classification

- Ideally, classify problems according to those that can be solved in polynomial-time and those that cannot.

- Provably requires exponential-time.
  - Given a board position in an n-by-n generalization of chess, can black guarantee a win?

- Provably undecidable.
  - Given a program and input there is no algorithm to decide if program halts.

- Frustrating news. Huge number of fundamental problems have defied classification for decades.

Polynomial-time Reductions

Instances

- A problem (problem type) is the general, abstract term:
  - Examples: Shortest Path, Maximum Flow, Closest Pair, Sequence Alignment, String Matching.

- A problem instance is the concrete realization of a problem.
  - Maximum flow. The instance consists of a flow network.
  - Closest Pair. The instance is a set of points
  - String Matching. The instance consists of two strings.

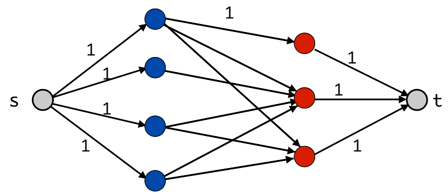
Polynomial-time reduction

- Reduction. Problem X polynomial reduces to problem Y if arbitrary instances of problem X can be solved using:
  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to oracle that solves problem Y.

- Notation. $X \leq_P Y$.
- We pay for time to write down instances sent to black box $\Rightarrow$ instances of Y must be of polynomial size.

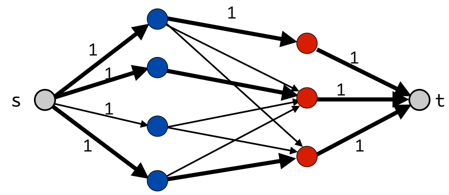
Maximum flow and bipartite matching

- Bipartite matching $\leq_P$ Maximum flow



Maximum flow and maximum bipartite matching

- Bipartite matching $\leq_P$ Maximum flow
  - Matching M => flow of value |M|
  - Flow of value v(f) => matching of size v(f)



Polynomial-time reductions

- Purpose. Classify problems according to relative difficulty.

  - Design algorithms. If X $\leq_P$ Y and Y can be solved in polynomial-time, then X can also be solved in polynomial time.

  - Establish intractability. If X $\leq_P$ Y and X cannot be solved in polynomial-time, then Y cannot be solved in polynomial time.

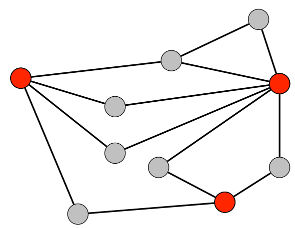
  - Establish equivalence. If X $\leq_P$ Y and Y $\leq_P$ X, we use notation X $=_P$ Y.

up to a
polynomial factor

Independent set and vertex cover

- Independent set: A set S of vertices where no two vertices of S are neighbors (joined by an edge).
- Independent set problem: Given graph G and an integer k, is there an independent set of size ≥ k?

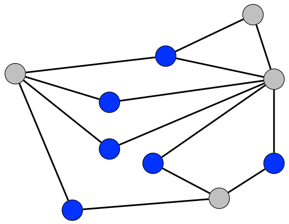
- Example:
  - Is there an independent set of size ≥ 6?



Independent set and vertex cover

- Independent set: A set S of vertices where no two vertices of S are neighbors (joined by an edge).
- Independent set problem: Given graph G and an integer k, is there an independent set of size ≥ k?

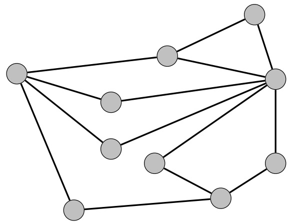
- Example:
  - Is there an independent set of size ≥ 6? Yes



Independent set and vertex cover

- Independent set: A set S of vertices where no two vertices of S are neighbors (joined by an edge).
- Independent set problem: Given graph G and an integer k, is there an independent set of size ≥ k?

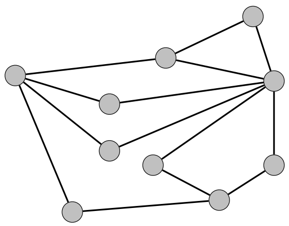
- Example:
  - Is there an independent set of size ≥ 6? Yes
  - Is there an independent set of size ≥ 7?



Independent set and vertex cover

- Independent set: A set S of vertices where no two vertices of S are neighbors (joined by an edge).
- Independent set problem: Given graph G and an integer k, is there an independent set of size ≥ k?

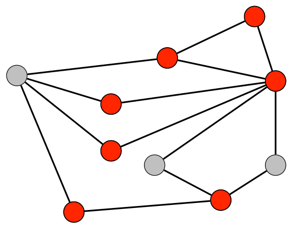
- Example:
  - Is there an independent set of size ≥ 6? Yes
  - Is there an independent set of size ≥ 7? No



Independent set and vertex cover

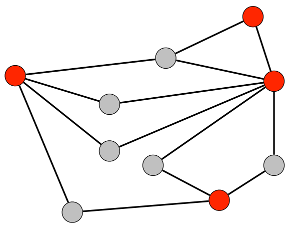
- Vertex cover: A set S of vertices such that all edges have at least one endpoint in S.
- Independent set problem: Given graph G and an integer k, is there a vertex cover of size ≤ k?

- Example:
  - Is there a vertex cover of size ≤ 4?



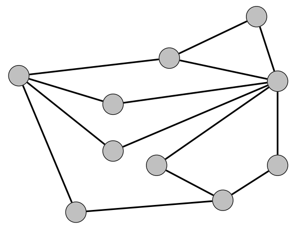
Independent set and vertex cover

- Vertex cover: A set S of vertices such that all edges have at least one endpoint in S.
- Independent set problem: Given graph G and an integer k, is there a vertex cover of size ≤ k?
- Example:
  - Is there a vertex cover of size ≤ 4? Yes



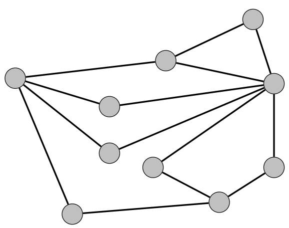
Independent set and vertex cover

- Vertex cover: A set S of vertices such that all edges have at least one endpoint in S.
- Independent set problem: Given graph G and an integer k, is there a vertex cover of size ≤ k?
- Example:
  - Is there a vertex cover of size ≤ 4? Yes
  - Is there a vertex cover of size ≤ 3?



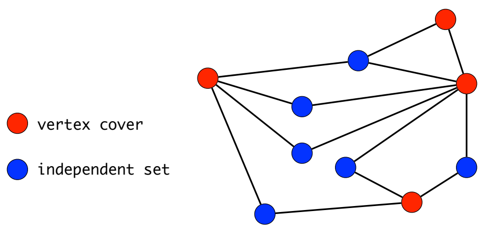
Independent set and vertex cover

- Vertex cover: A set S of vertices such that all edges have at least one endpoint in S.
- Independent set problem: Given graph G and an integer k, is there a vertex cover of size ≤ k?
- Example:
  - Is there a vertex cover of size ≤ 4? Yes
  - Is there a vertex cover of size ≤ 3? No



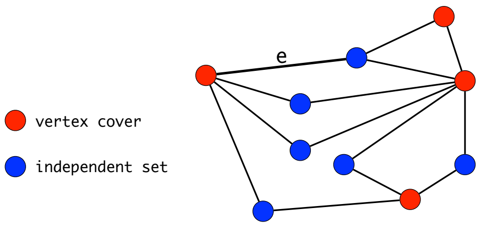
Independent set and vertex cover

- Claim. Let G=(V,E) be a graph. Then S is an independent set if and only if its complement V-S is a vertex cover.
- Proof.
  - =>: S is an independent set.



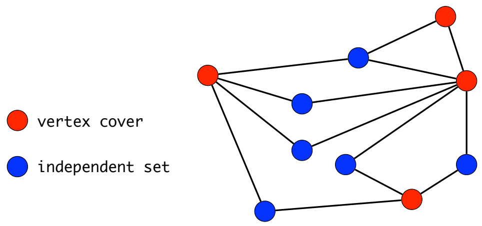
Independent set and vertex cover

- Claim. Let G=(V,E) be a graph. Then S is an independent set if and only if its complement V-S is a vertex cover.
- Proof.
  - =>: S is an independent set.
    - e cannot have both endpoints in S => e have an endpoint in V-S.
    - V-S is a vertex cover.



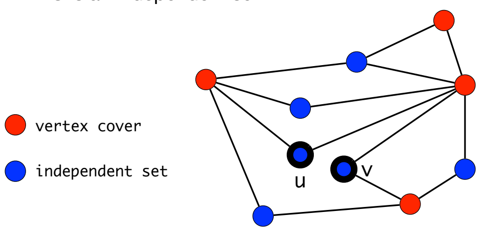
Independent set and vertex cover

- Claim. Let G=(V,E) be a graph. Then S is an independent set if and only if its complement V-S is a vertex cover.
- Proof.
  - =>: S is an independent set.
    - e cannot have both endpoints in S => e have an endpoint in V-S.
    - V-S is a vertex cover
  - <=: V-S is a vertex cover.



Independent set and vertex cover

- Claim. Let G=(V,E) be a graph. Then S is an independent set if and only if its complement V-S is a vertex cover.
- Proof.
  - =>: S is an independent set.
    - e cannot have both endpoints in S => e have an endpoint in V-S.
    - V-S is a vertex cover
  - <=: V-S is a vertex cover.
    - u and v not part of the vertex cover = > no edge between u and v
    - S is an independent set.



Independent set and vertex cover

- Claim. Let G=(V,E) be a graph. Then S is an independent set if and only if its complement V-S is a vertex cover.

- Independent set $\leq_P$ vertex cover
  - Use one call to the black box vertex cover algorithm with k = n-k.
  - There is an independent set of size ≥ k if and only if the vertex cover algorithm returns yes.

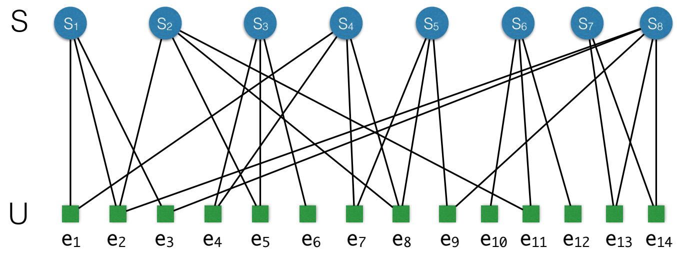
- vertex cover $\leq_P$ independent set
  - Use one call to the black box independent set algorithm with k = n-k.

Set cover

- Set cover. Given a set U of elements, a collection of sets $S_1, \ldots S_m$ of subsets of U, and an integer k. Does there exist a collection of at most k sets whose union is equal to all of U?

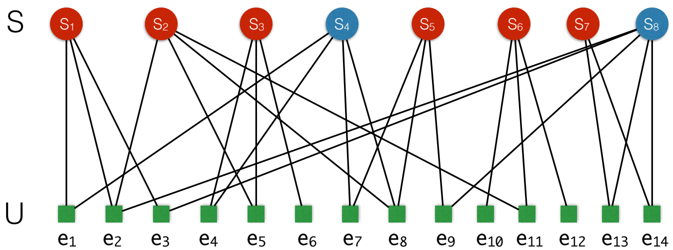
Set cover

- Set cover. Given a set U of elements, a collection of sets $S_1, \ldots S_m$ of subsets of U, and an integer k. Does there exist a collection of at most k sets whose union is equal to all of U?
- Example:
  - Does there exist a set cover of size at most 6?



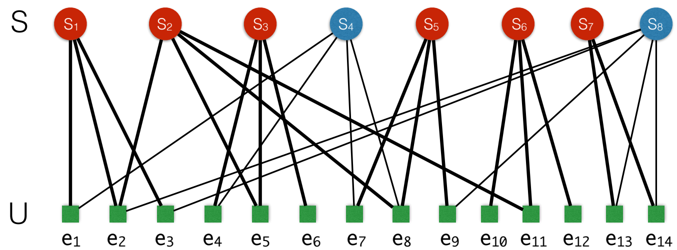
Set cover

- Set cover. Given a set U of elements, a collection of sets $S_1, \ldots S_m$ of subsets of U, and an integer k. Does there exist a collection of at most k sets whose union is equal to all of U?
- Example:
  - Does there exist a set cover of size at most 6? Yes



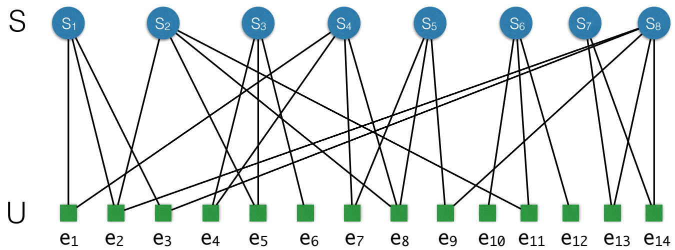
Set cover

- Set cover. Given a set U of elements, a collection of sets $S_1, \ldots S_m$ of subsets of U, and an integer k. Does there exist a collection of at most k sets whose union is equal to all of U?
- Example:
  - Does there exist a set cover of size at most 6? Yes



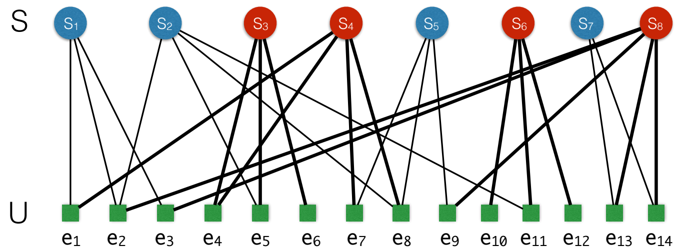
Set cover

- Set cover. Given a set U of elements, a collection of sets $S_1, \ldots S_m$ of subsets of U, and an integer k. Does there exist a collection of at most k sets whose union is equal to all of U?
- Example:
  - Does there exist a set cover of size at most 6? Yes
  - Does there exist a set cover of size at most 4?



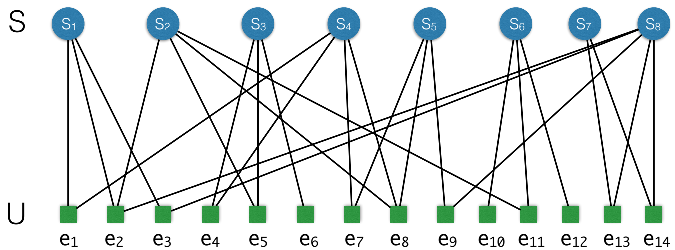
Set cover

- Set cover. Given a set U of elements, a collection of sets $S_1, \ldots S_m$ of subsets of U, and an integer k. Does there exist a collection of at most k sets whose union is equal to all of U?
- Example:
  - Does there exist a set cover of size at most 6? Yes
  - Does there exist a set cover of size at most 4? Yes



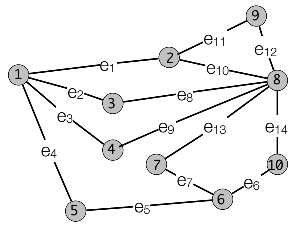
Set cover

- Set cover. Given a set U of elements, a collection of sets $S_1, \ldots S_m$ of subsets of U, and an integer k. Does there exist a collection of at most k sets whose union is equal to all of U?
- Example:
  - Does there exist a set cover of size at most 6? Yes
  - Does there exist a set cover of size at most 4? Yes
  - Does there exist a set cover of size at most 3? No



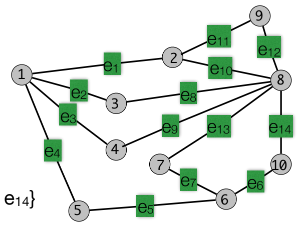
Reduction from vertex cover to set cover

- vertex cover $\leq_P$ set cover



Reduction from vertex cover to set cover

- vertex cover $\leq_P$ set cover
- U = {e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14},}
- S_1 = {e_1, e_2, e_3, e_4}
- S_2 = {e_1, e_{11}, e_{10}}
- S_3 = {e_2, e_8}
- S_4 = {e_3, e_9}
- S_5 = {e_4, e_5}
- S_6 = {e_5, e_6, e_7}
- S_7 = {e_7, e_{13}}
- S_8 = {e_8, e_9, e_{10}, e_{12}, e_{13}, e_{14}}
- S_9 = {e_{11}, e_{12}}
- S_{10} = {e_6, e_{14}}



Polynomial-time reductions

- Reduction. X $\leq_P$ Y if arbitrary instances of problem X can be solved using:
  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to oracle that solves problem Y.

- Strategy to make a reduction if we only need one call to the oracle/black box to solve X:
  1. Show how to turn (any) instance S_x of X into an instance of S_y of Y in polynomial time.
  2. Show that: S_x a yes instance of X => S_y a yes instance of Y.
  3. Show that: S_y a yes instance to Y => S_x a yes instance of X.

- Reductions that needs more than one call to black box:
  1. Show how to turn (any) instance S_x of X into a polynomial number instance of $S_{y,i}$ of Y in polynomial time.
  2. Show: S_x a yes instance of X => one of the instances $S_{y,i}$ is a yes instance of Y.
  3. Show: one of the instances $S_{y,i}$ is a yes instance of Y => S_x a yes instance of X.

P and NP

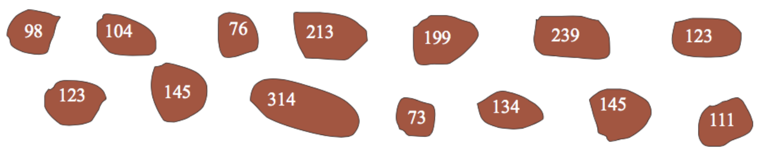
The class P

- P ~ problems solvable in deterministic polynomial time.
  - Given a problem type X, there is a deterministic algorithm *A* which for every problem instance *I* ∈ X solves *I* in a time that is polynomial in |*I*|, the size of *I*.
  - I.e., the running time of *A* is $O(|I|^k)$ for all *I* ∈ X, where k is constant independent of the instance *I*.

- Examples.
  - Closest pair: There is an algorithm A such that for every set S of points, A finds a closest pair in time $O(|S|^2)$.
  - Maximum flow: There is an algorithm A such that for any network, A finds a maximum flow in time $O(|V|^3)$, where V is the set of vertices.

Hard problems: Example

- Problem [POTATO SOUP]. A recipe calls for B grams of potatoes. You have a K kilo bag with n potatoes. Can one select some of them such that their weight is exactly B grams?



- Best known solution: create all 2^n subsets and check each one.

Hard problems

- Many problems share the above features
  - Can be solved in time $2^{|T|}$ (by trying all possibilities.)
  - Given a potential solution, it can be checked in time $O(|I|^k)$, whether it is a solution or not.
- These problems are called polynomially checkable.
- A solution can be guessed, and then verified in polynomial time.

The class NP

- Certifier. Algorithm B(s,t) is an **efficient certifier** for problem X if:
  1. B(s,t) runs in polynomial time.
  2. For every instance s: *s is a yes instance of X*

$$\Leftrightarrow$$

there exists a certificate t of length polynomial in s and *B(s,t) returns yes.*

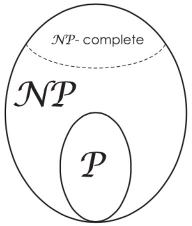
- Example. Independent set.
  - s: a graph G and an integer k.
  - t: a set of vertices from G.
  - B(s,t) returns yes if and only if t is an independent set of G and $|S| \geq k$.
  - This can be checked in polynomial time by checking that no two vertices in t are neighbors and that the size is at least k.
- A problem X is in the class NP (Non-deterministic Polynomial time) if X has an efficient certifier.

Optimization vs decision problems

- Consider decision problems (yes-no-problems).
- Example.
  - [POTATO SOUP]. A recipe calls for B grams of potatoes. You have a K kilo bag with n potatoes. Can one select some of them such that their weight is exactly B grams?
- Optimization vs decision problem
  - [OPTIMIZATION LONGEST PATH] Given a graph G. What is the length of the longest simple path?
  - [DECISION LONGEST PATH] Given a graph G and integer k. Is a there a simple path of length $\geq$ k?
- Exercise. Show that OPTIMIZATION LONGEST PATH can be solved in polynomial time if and only if DECISION LONGEST PATH can be solved in polynomial time.

P vs NP

- P solvable in deterministic polynomial time.
- NP solvable in non-deterministic polynomial time/ has an efficient (polynomial time) certifier.
- P⊆NP (every problem T which is in P is also in NP).
- It is not known (but strongly believed) whether the inclusion is proper, that is whether there is a problem in NP which is not in P.
- There is subclass of NP which contains the hardest problems, NP-complete problems:
  - X is NP-Complete if
    - X ∊ NP
    - $Y \leq_P X$ for all Y ∊ NP



Examples of NP-complete problems

- Preparing potato soup
- Packing your suitcase
- Satisfiability of clauses
- Partition
- Subset-sum
- Hamilton Cycle
- Travelling Salesman
- Bin Packing
- Knapsack
- Clique
- Independent Set
- Vertex Cover
- Set Cover

NP-complete problems

- [SOCCER CHAMPIONSHIP 3-POINT RULE] In a football league n teams compete for the championship. The leagues uses the 3-point rule, i.e., the points of match are distributed as 3:0, 1:1, or 0:3.
  - **Input.** The table with the points of every team at some point in the season, a list of the matches to be played in that season and the name of some team.
  - **Output.**
    - YES if the named team still can become champion
    - NO otherwise.

NP-complete problems

- [SATISFIABILITY]
  - **Input:** A set of clauses C = {c1, . . . , ck} over n boolean variables x1,…,xn.
  - **Output:**
    - YES if there is a satisfying assignment, i.e., if there is an assignment a: $\{x_1,\ldots,x_n\} \rightarrow \{0,1\}$ such that every clause is satisfied,
    - NO otherwise.

$$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$$

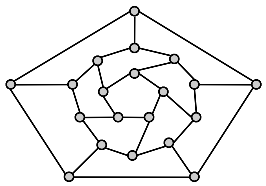
instance s

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

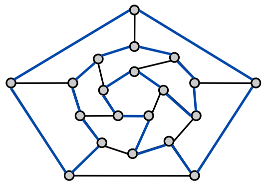
proposed solution/certificate t

NP-complete problems

- [HAMILTONIAN CYCLE].
  - **Input:** Undirected graph G
  - **Output:**
    - YES if there exists a simple cycle that visits every node
    - NO otherwise

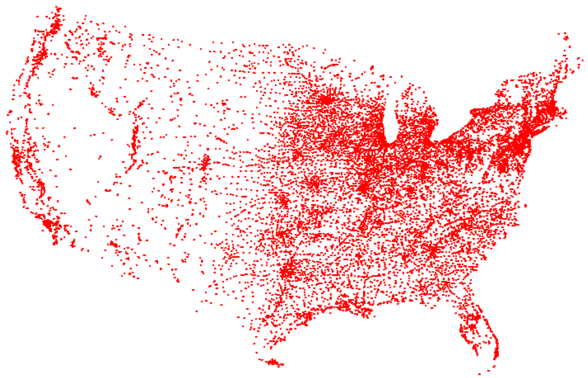


instance s



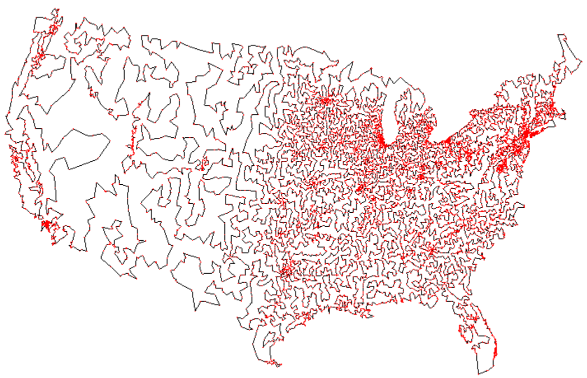
certificate t

- Traveling Salesperson Problem TSP: Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?



All 13,509 cities in US with a population of at least 500
Reference: http://www.tsp.gatech.edu

- Traveling Salesperson Problem TSP: Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?



Optimal TSP tour
Reference: http://www.tsp.gatech.edu

How to prove a problem is NP-complete

1. Prove $Y \in NP$ (that it can be verified in polynomial time).
2. Select a known NP-complete problem X.
3. Give a polynomial time reduction from X to Y (prove $X \leq_P Y$):
   - Explain how to turn an instance of X into one or more instances of Y
   - Explain how to use a polynomial number of calls to the black box algorithm/oracle for Y to solve X.
   - Prove/argue that the reduction is correct.

Reduction example

- [HAMILTONIAN CYCLE]. Given a undirected graph G=(V,E), does there exists a simple cycle that visits every node?
- [TRAVELLING SALESMAN (TSP)] Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?

- Show Hamiltonian Cycle $\leq_P$ TSP:
  - Idea: For every instance of Hamiltonian Cycle create an instance of TSP such that the TSP instance has tour of length ≤ n if and only if G has a Hamiltonian cycle.

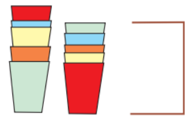
- Reduction.
  - Given instance G=(V,E) of Hamiltonian Cycle, create n cities with distance function

$$d(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 2 & \text{if } (u, v) \notin E \end{cases}$$

  - TSP instance has tour of length ≤ n if and only if G has a Hamiltonian cycle.

Reduction example

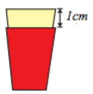
- [GLASSES IN A CUPBOARD]. You have n glasses of equal height. If glass g_j is put into glass g_i let d_{ij} be the amount of g_j above the rim of g_i. You want to stack them into a single stack, so they fit into a cupboard of height h; is that possible?



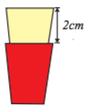
- Glasses in a Cupboard in NP: Proposed solution can be verified in polynomial time.
- NP-completeness:
  - Reduction from Directed Hamiltonian Path (DHP).
  - Directed Hamiltonian Path: Given a directed graph G, is there a directed simple path visiting all vertices.
  - DHP is NP-complete
  - Reduction: For every instance (graph) of DHP make a set of glasses and a cupboard, such that the glasses can be stacked into the cupboard if and only if the graph has a Hamiltonian path.

Reduction example

- Let G = (E, V) a directed graph.
  - Make one glass for every node i ∈ V.
  - If (i, j) ∈ E ensure:

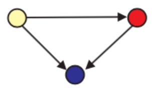


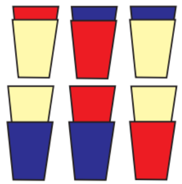
  - If (i, j) ∉ E ensure:

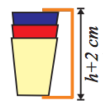


  - Glass i is red, glass j is yellow.
  - Height of the cupboard is |V| – 1 + height of glass

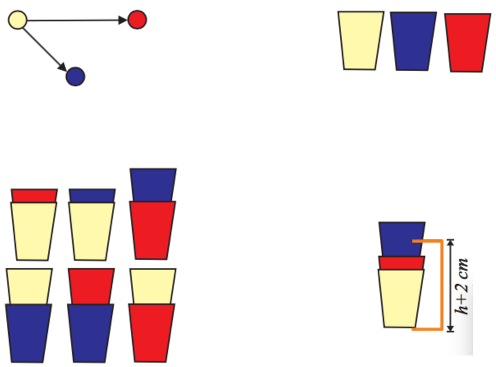
Reduction Example







Reduction Example



The Main Question: P Versus NP

- Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
  - Is the decision problem as easy as the certification problem?
  - Clay $1 million prize.

- Consensus opinion on P = NP? Probably no.