P and NP

Inge Li Gørtz

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

Hardness of problems

- Want to understand how difficult or easy a given problem is.
  - Know there are problems that can be solved in polynomial time (all problems seen in this course). Easy
  - There are problems we cannot solve! Unsolvable
  - What about in between?

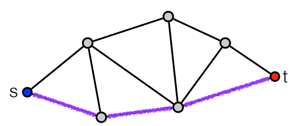
Problem Classification

- 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]

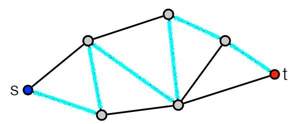
Problem Classification

- 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)
2-coloring	3-coloring



Shortest s-t path?

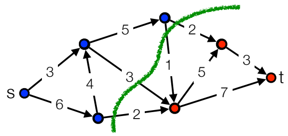


Longest s-t path?

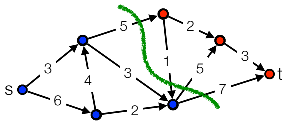
Problem Classification

- 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)
2-coloring	3-coloring



Minimum s-t cut?



Maximum s-t cut?

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)
2-coloring	3-coloring

1. DIVISION 1990 < 1989 | 1991 >

SAMLET HJEMME UDE EFTERÅR FORÅR

#	KLUBNAVN	K	V	U	T	SCORE	+/-	POINT
1	Brøndby IF	26	17	8	1	50- 16	+34	42
2	B 1903	26	10	11	5	44- 27	+17	31
3	Ikast FS	26	11	8	7	38- 27	+11	30
4	Silkeborg IF	26	11	8	7	35- 26	+9	30
5	BK Frem	26	7	15	4	33- 25	+8	29
6	Lyngby BK	26	10	8	8	44- 30	+14	28
7	AGF	26	9	10	7	31- 25	+6	28

SUPERLIGAEN (3F SUPERLIGAEN) 2020/21 < 2019/20 |

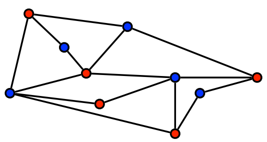
SAMLET HJEMME UDE EFTERÅR FORÅR *Opdateret til og med 04.10.2020*

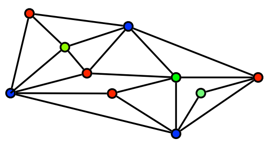
#	KLUBNAVN	K	V	U	T	SCORE	+/-	POINT
1	Brøndby IF	4	4	0	0	9- 5	+4	12
2	AGF	4	2	2	0	10- 6	+4	8
3	Vejle BK (O)	4	2	1	1	9- 7	+2	7
4	SønderjyskE	4	2	1	1	8- 7	+1	7
5	FC Midtjylland (M)	4	2	1	1	4- 4	0	7
6	AaB	4	1	2	1	3- 4	-1	5
7	FC Nordsjælland	4	1	1	2	9- 8	+1	4

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)
2-coloring	3-coloring


2-coloring?


3-coloring?

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.

Overview

- Reductions
  - Tools for classifying problems according to relative hardness
- P and NP

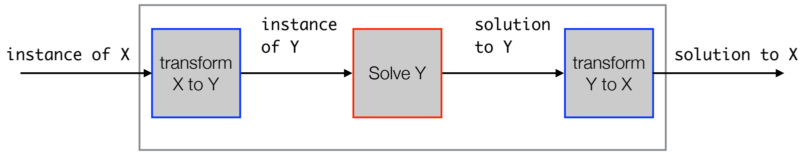
Polynomial-time Reductions

Instances

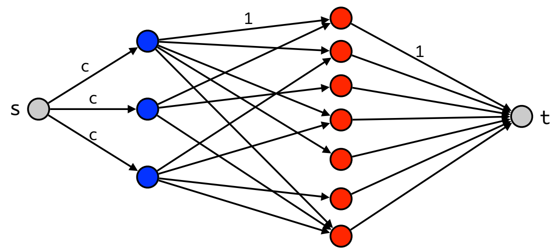
- A problem (problem type) is the general, abstract term:
  - Examples: Shortest Path, Maximum Flow, Closest Pair, Sequence Alignment, String Matching.
- A problem instance is the concrete realization of a problem.
  - Maximum flow. The instance consists of a flow network.
  - Shortest path. The instance is a graph.
  - String Matching. The instance consists of two strings.

Polynomial-time reduction

- Reduction from problem X to problem Y.

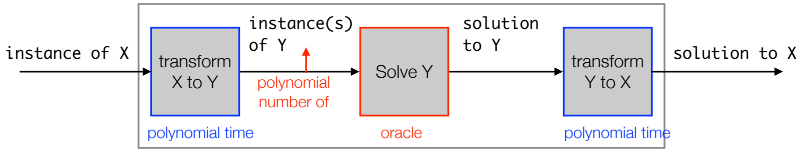


- Example. Scheduling of doctors.



Polynomial-time reduction

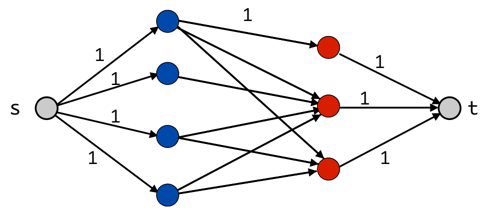
- Reduction from problem X to problem Y.



- Reduction. Problem X polynomial reduces to problem Y if any instance 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.

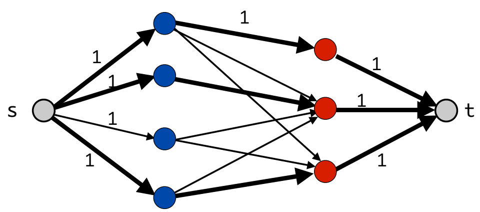
Maximum flow and bipartite matching

- 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

Polynomial-time reductions

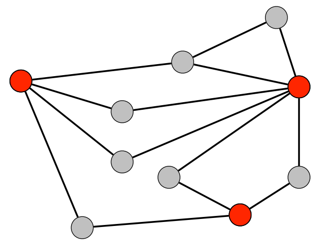
- Reduction. $X \leq_P Y$ if arbitrary instances of problem X can be solved using:
  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to oracle that solves problem Y.

- 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.

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?

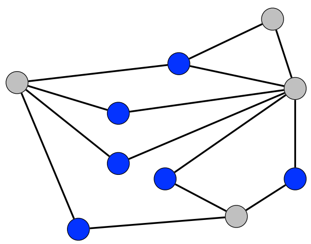
- Example:
  - Is there an independent set of size ≥ 6?



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?

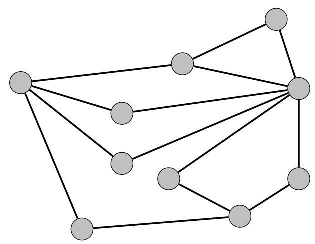
- Example:
  - Is there an independent set of size ≥ 6? Yes



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?

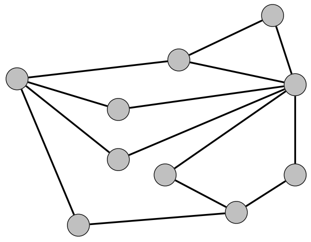
- Example:
  - Is there an independent set of size ≥ 6? Yes
  - Is there an independent set of size ≥ 7?



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?

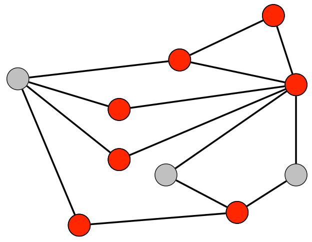
- Example:
  - Is there an independent set of size ≥ 6? Yes
  - Is there an independent set of size ≥ 7? No



Independent set and vertex cover

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

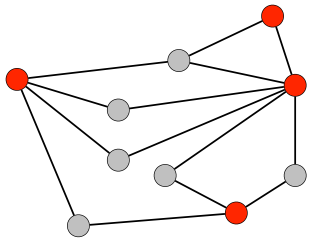
- Example:
  - Is there a vertex cover of size ≤ 4?



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?

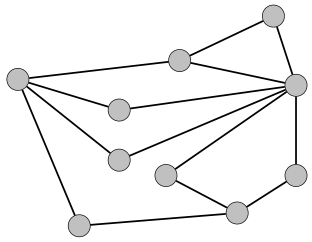
- Example:
  - Is there a vertex cover of size ≤ 4? Yes



Independent set and vertex cover

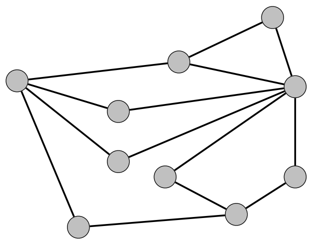
- Vertex cover: A set S of vertices such that all edges have at least one endpoint in S.
- Independent set problem: Given graph G and an integer k, is there a vertex cover of size ≤ k?

- Example:
  - Is there a vertex cover of size ≤ 4? Yes
  - Is there a vertex cover of size ≤ 3?



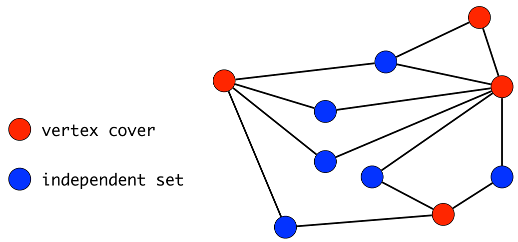
Independent set and vertex cover

- 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



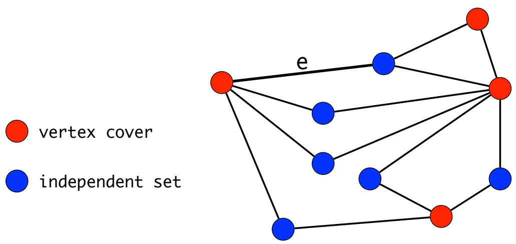
Independent set and vertex cover

- 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.



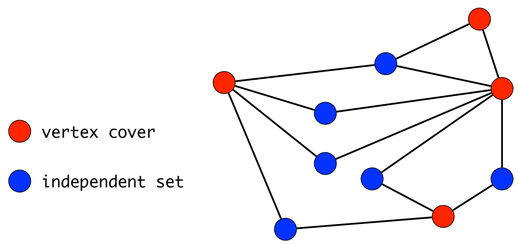
Independent set and vertex cover

- 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.



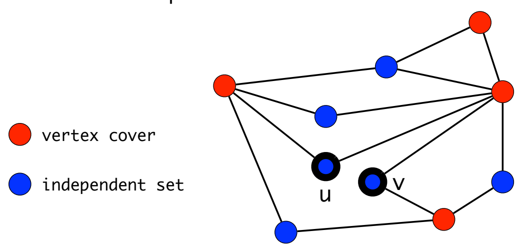
Independent set and vertex cover

- 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.

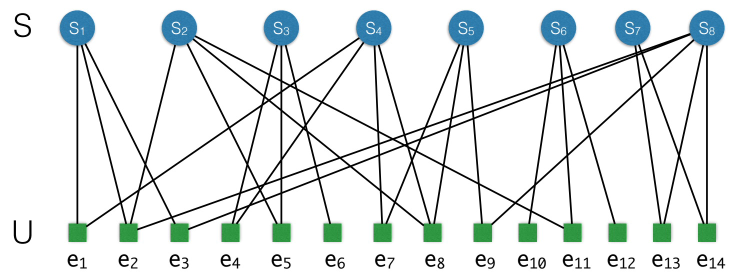
- vertex cover $=_P$ independent set

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?

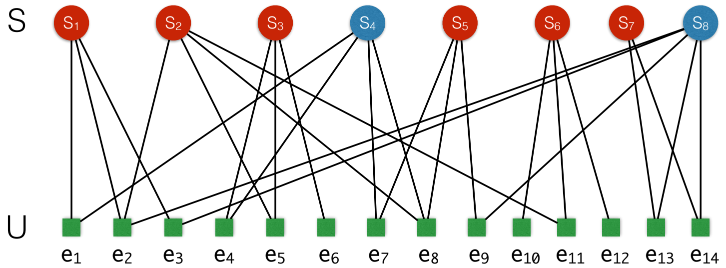
Set cover

- 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?



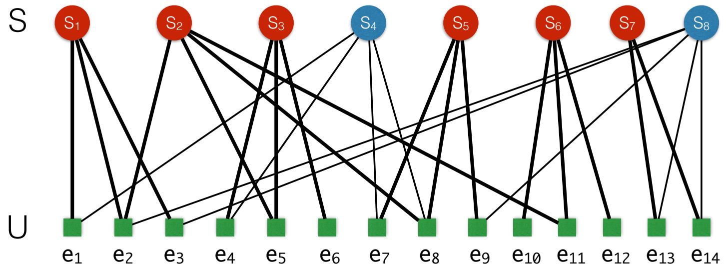
Set cover

- 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



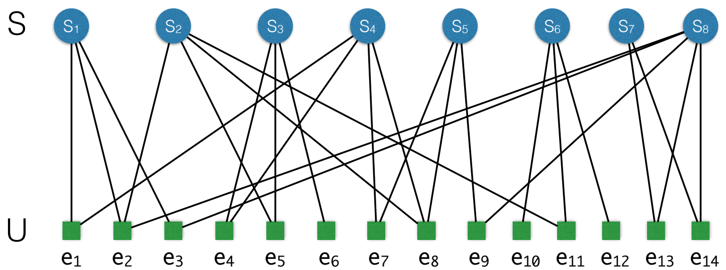
Set cover

- 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



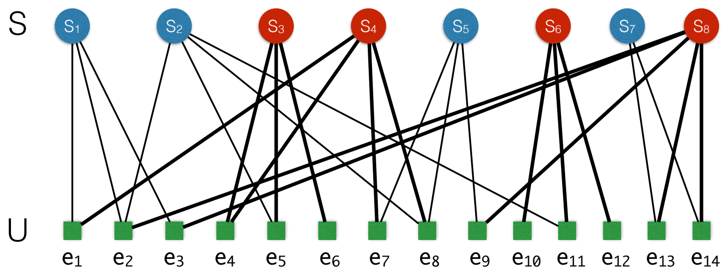
Set cover

- 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?



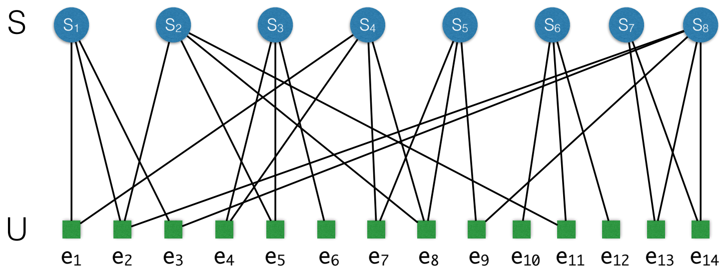
Set cover

- 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



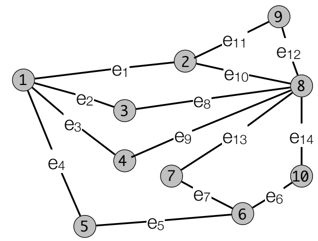
Set cover

- 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



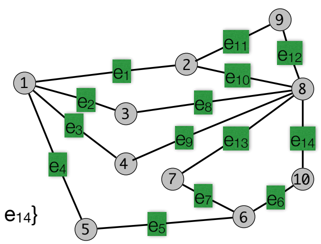
Reduction from vertex cover to set cover

- vertex cover $\leq_P$ set cover



Reduction from vertex cover to set cover

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



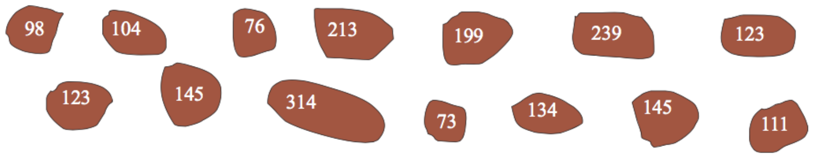
P and NP

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 $I \in X$ solves I in a time that is polynomial in $|I|$, the size of I.
  - lRunning time of *A* is $O(|I|^k)$ for all $I \in X$, where k is a constant independent of the instance I.
- Examples.
  - Maximum flow: There is an algorithm A that for any network finds a maximum flow in time $O(|V|^3)$, where V is the set of vertices.
  - String matching: There is an algorithm A that for any text T and pattern P finds all occurrences of P in T in $O(|P| + |T|)$ time.

Hard 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?



- 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.

Optimization vs decision problems

- 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.

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.*

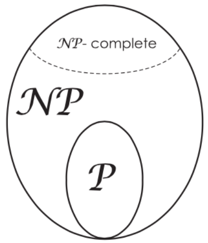
     proposed solution

- Example. Independent set.
  - s: a graph G and an integer k.
  - t: a set of vertices from G.
  - B(s,t) returns yes $\Longleftrightarrow$ t is an independent set of G and |S| ≥ k.
  - Check in polynomial time: check that no two vertices in t are neighbors and that the size is at least k.

- NP. A problem X is in the class NP (Non-deterministic Polynomial time) if X has an efficient certifier.

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).
- P = NP?
- 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 (Subset Sum)
- Independent Set
- Vertex Cover
- Set Cover
- Longest path
- Max cut
- Soccer championship (3-point rule)
- 3-coloring

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: $\{x_1,\ldots,x_n\}$ [?]→ {0,1} such that every clause is satisfied,
    - NO otherwise.

$$(\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (\overline{x_1} \lor \overline{x_3} \lor \overline{x_4})$$

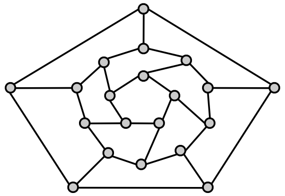
instance s

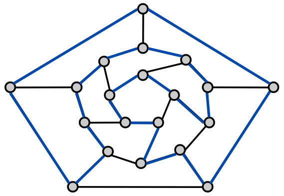
$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

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

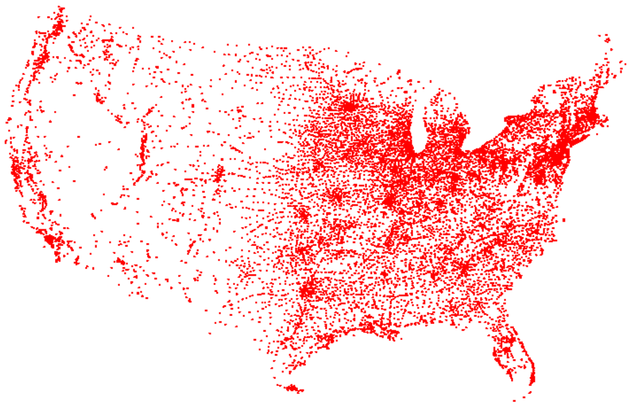

instance s


certificate t

How to prove a problem is NP-complete

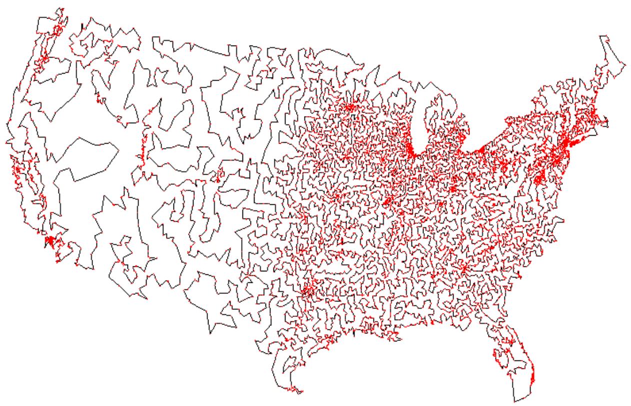
1. Prove $Y \in NP$ (that it can be verified in polynomial time).
2. Select a known NP-complete problem X.
3. Give a polynomial time reduction from X to Y (prove $X \leq_P Y$):
   - Explain how to turn an instance of X into one or more instances of Y
   - Explain how to use a polynomial number of calls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

- 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

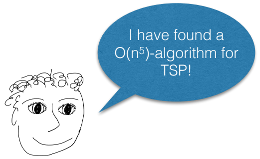
- Traveling Salesperson Problem (TSP): Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?

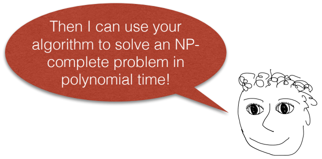


Optimal TSP tour
Reference: http://www.tsp.gatech.edu

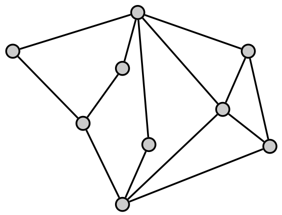
Reduction example: TSP

- There is no polynomial time algorithm for TSP for unless P=NP.

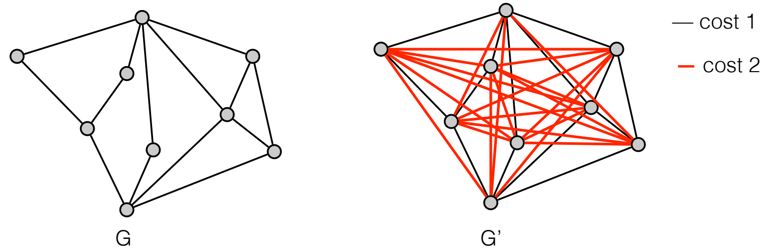




- *Hamiltonian cycle.* Given G=(V,E). Is there a cycle visiting every vertex exactly once?

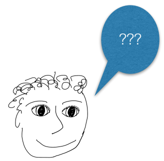


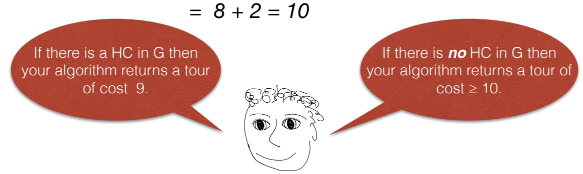
Hamiltonian Cycle ≤P TSP



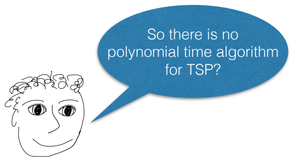
- *G has a Hamiltonian cycle* ⇔ *optimal cost of TSP in G' is n = 9.*
- *G has no Hamiltonian cycle* ⇔ *optimal cost of TSP in G' is at least n -1 + 2*

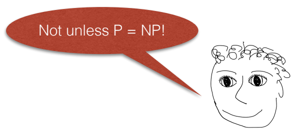
= 8 + 2 = 10





TSP: Hardness





- TSP is NP-complete:
  - *Hamiltonian cycle* ≤P TSP.
  - TSP ∈ NP.
    - Certificate: Tour given as list of nodes $v_1, v_2, \ldots, v_n$.
    - Certifier: Check that
      - there is an edge from v_i to v_{i+1}
      - all nodes are in the list.

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.

The Simpsons: P = NP?