Time Complexity

- We use a multitape Turing machine

- We count the number of steps until a string is accepted

- We use the $O(k)$ notation

Example: $L = \{a^n b^n : n \geq 0\}$

Algorithm to accept a string w :

- Use a two-tape Turing machine
- Copy the a on the second tape
- Compare the a and b

$$L = \{a^n b^n : n \geq 0\}$$

Time needed:

- Copy the a on the second tape $O(|w|)$
- Compare the a and b $O(|w|)$

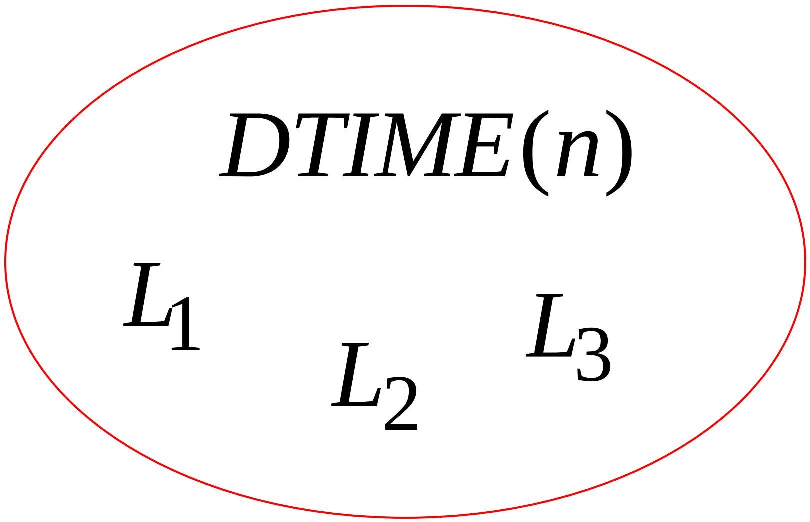
Total time: $O(|w|)$

$$L = \{a^n b^n : n \geq 0\}$$

For string of length n

time needed for acceptance: $O(n)$

Language class: $DTIME(n)$



A Deterministic Turing Machine accepts each string of length n in time $O(n)$

$DTIME(n)$

$\{a^n b^n : n \geq 0\}$

$\{ww\}$

In a similar way we define the class

$$DTIME(T(n))$$

for any time function: $T(n)$

Examples: $DTIME(n^2), DTIME(n^3), \ldots$

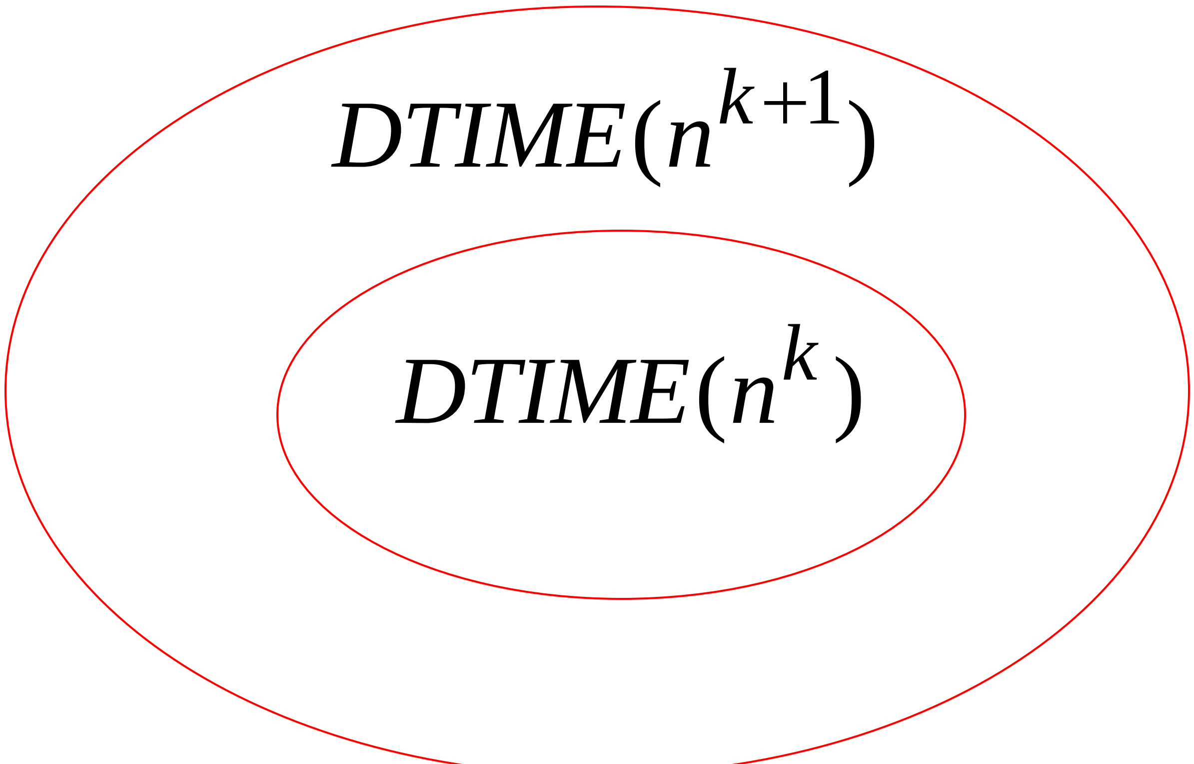
Example: The membership problem for context free languages

$$L = \{w : w \text{ is generated by grammar } G\}$$

$$L \in DTIME(n^3)$$

Polynomial time

Theorem: $DTIME(n^{k+1}) \supset DTIME(n^{k})$



Polynomial time algorithms: $DTIME(n^k)$

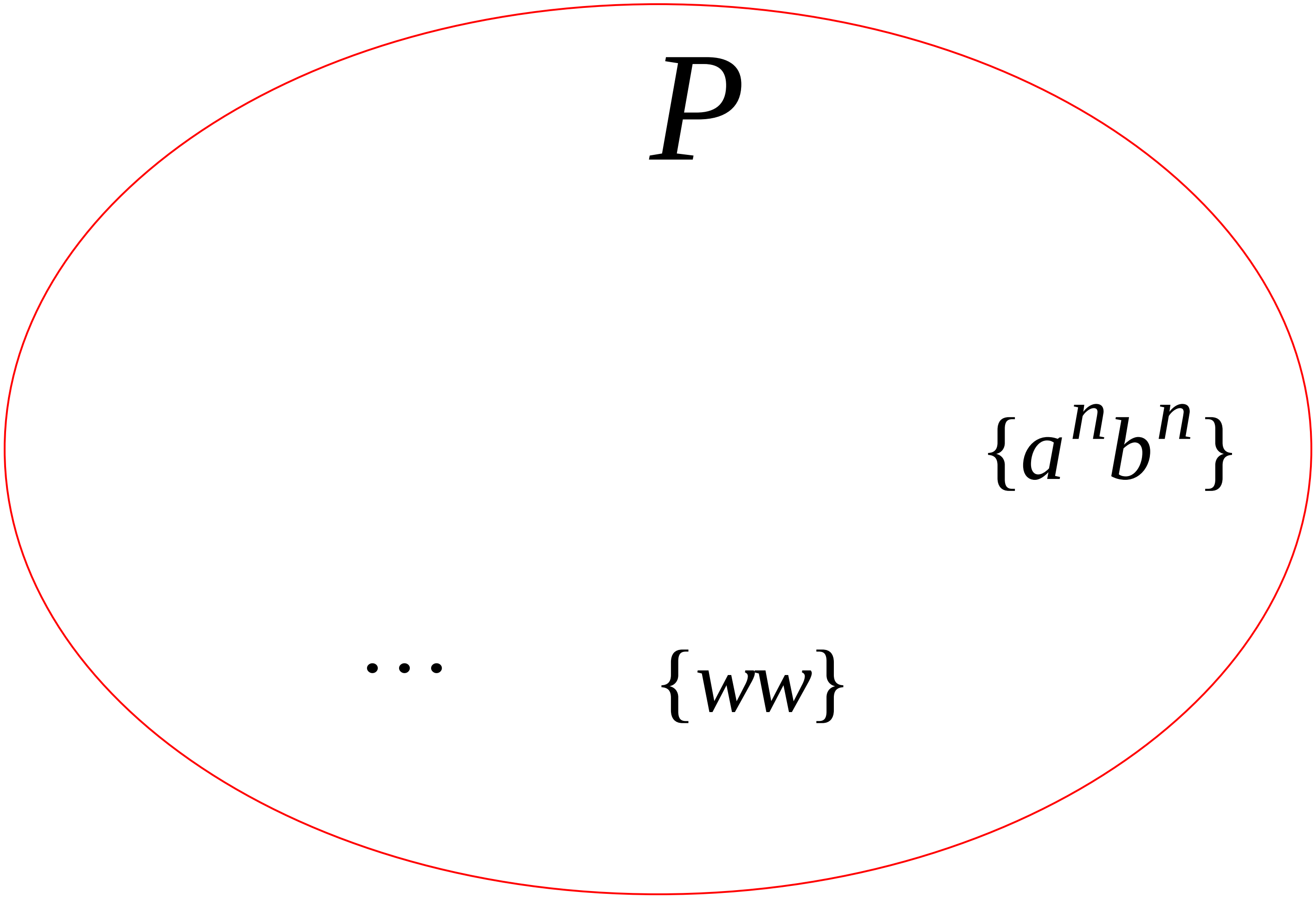
Represent tractable algorithms:

For small k we can compute the result fast

The class P

$$P = \cup DTIME(n^k) \qquad \text{for all } k$$

- Polynomial time
- All tractable problems


P
$\{a^n b^n\}$
$\cdots$
$\{ww\}$

Examples of problems in P

1, PATH={<G,s,t>|G is a directed graph that has a directed path from s to t}

- The PATH problem: Is a path from s to t?

2, REPRIME={<x,y>|x and y are relatively prime}

- The REPRIME problem: Are x and y relative primes?

PATH

A polynomial time algorithm M of PATH operates as follows.

M = "On input<G, s, t> where G is a directed graph with nodes s and t:

1. Place a mark on node s.

2. Repeat the following until no additional nodes are marked:

PATH

3. Scan all the edges of G. If an edge (a, b) is found going from a marked node a to an unmarked node b, mark node b.

4. If t is marked, *accept*. Otherwise, *reject*."

REPRIME

The Euclidean algorithm E is as follows.

E = "On input <x, y>, where x and y are natural numbers in binary:

1. Repeat until $y = 0$:
2. Assign $x \leftarrow x \bmod y$.
3. Exchange x and y.
4. Output x."

REPRIME

Algorithm R solves RELPRIME, using E as a subroutine.

R = "On input <x, y>, where x and y are natural numbers in binary:

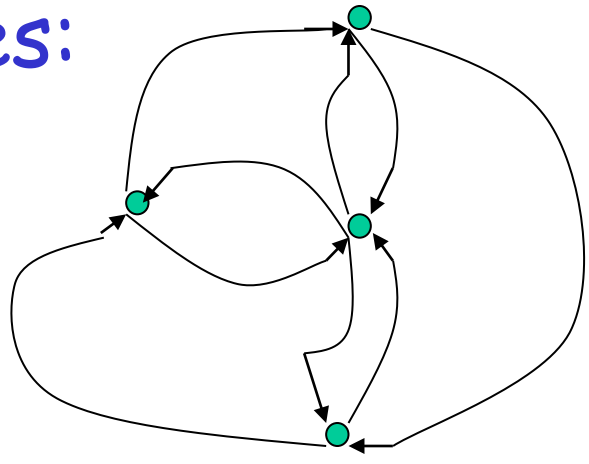
1. Run E on <x, y>.

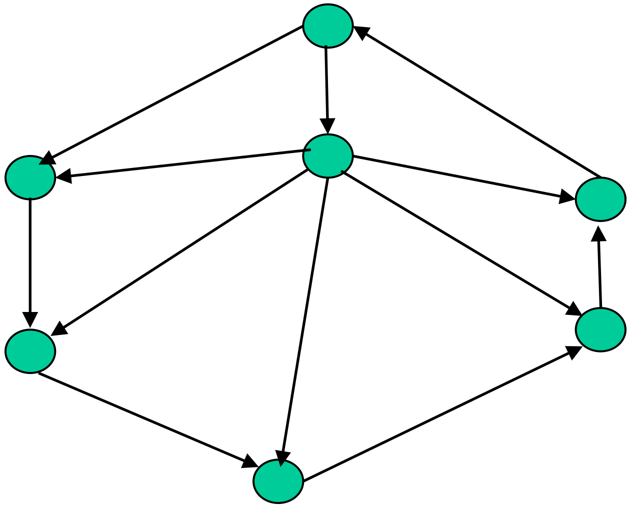
2. If the result is 1, *accept*. Otherwise, *reject*."

Eulerian Graphs

- Euler Cycle Problem: Given a graph G, is there a closed path in G that uses each edge exactly once?
- A graph that has a Euler cycle is called Eulerian.

Examples:





Eulerian Graphs

Euler Cycle Problem is in P. That is,

$L = \{Cycle(G) \mid G \quad is \quad Eulerian\}$ is in P.

Proof. By Euler Theorem:

Euler Theorem: A graph G is Eulerian iff the following two conditions are true:

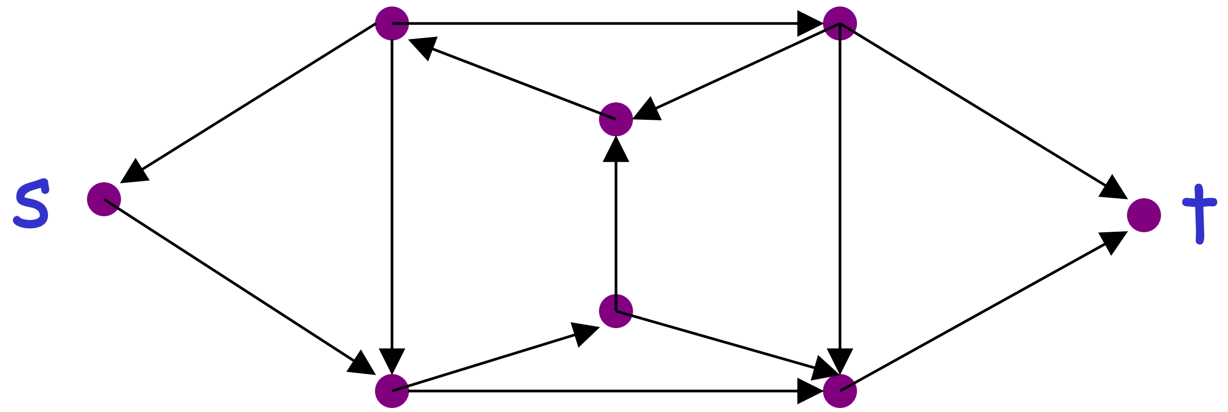
- For any pair of nodes u, v in G, there is a path from u to v.
- All nodes have equal numbers of incoming and outgoing edges

Exponential time algorithms: $DTIME(2^n)$

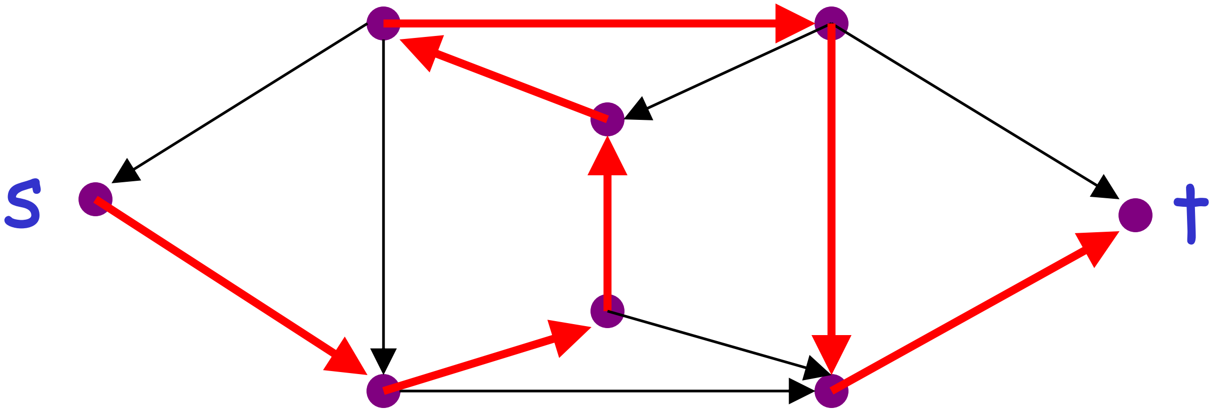
Represent intractable algorithms:

Some problem instances may take centuries to solve

Example: the Hamiltonian Problem



Question: is there a Hamiltonian path from s to t?



YES!

A solution: search exhaustively all paths

L = {<G,s,t>: there is a Hamiltonian path in G from s to t}

$$L \in DTIME(n!) \approx DTIME(2^n)$$

Sterling equation

Exponential time

Intractable problem

Example: The Satisfiability Problem (SAT)

SAT={<Φ>|Φ is a satisfiable Boolean formula.}

Boolean expressions in
Conjunctive Normal Form:

$$t_1 \wedge t_2 \wedge t_3 \wedge \cdots \wedge t_k$$

$$t_i = x_1 \vee \overline{x}_2 \vee x_3 \vee \cdots \vee \overline{x}_p$$

Variables

Question: is expression satisfiable?

Example: $$(\overline{x}_1 \vee x_2) \wedge (x_1 \vee x_3)$$

Satisfiable: $$x_1 = 0,\ \ x_2 = 1,\ \ x_3 = 1$$

$$(\overline{x}_1 \vee x_2) \wedge (x_1 \vee x_3) = 1$$

Example: $(x_1 \vee x_2) \wedge \overline{x}_1 \wedge \overline{x}_2$

Not satisfiable

SAT={<Φ>|Φ is a satisfiable Boolean formula.}

For n variables: $L \in DTIME(2^n)$

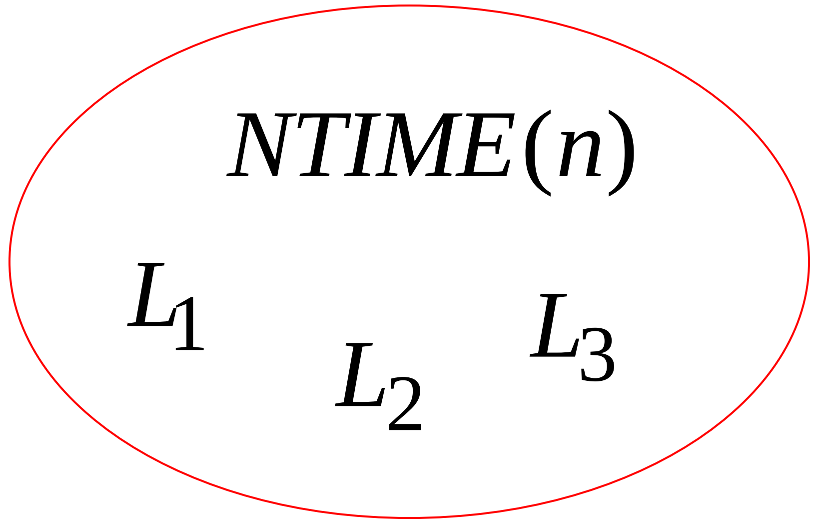
exponential

Algorithm:

search exhaustively all the possible binary values of the variables

Non-Determinism

Language class: $NTIME(n)$



A Non-Deterministic Turing Machine accepts each string of length n in time $O(n)$

Example: $L = \{ww\}$

Non-Deterministic Algorithm
to accept a string ww :

- Use a two-tape Turing machine

- Guess the middle of the string
  and copy w on the second tape

- Compare the two tapes

$$L = \{ww\}$$

Time needed:

- Use a two-tape Turing machine
- Guess the middle of the string and copy w on the second tape $O(|w|)$
- Compare the two tapes $O(|w|)$

Total time: $O(|w|)$

$$NTIME(n)$$

$$L = \{ww\}$$

In a similar way we define the class

$$NTIME(T(n))$$

for any time function: $T(n)$

Examples: $NTIME(n^2), NTIME(n^3), \ldots$

Non-Deterministic Polynomial time algorithms:

$$L \in NTIME(n^k)$$

The class NP

$$NP = \cup NTIME(n^k) \quad \text{for all } k$$

Non-Deterministic Polynomial time

Theorem 7.11 Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time nondeterministic single-tape Turing machine has an equivalent $2^{O(t(n))}$ time deterministic single-tape Turing machine.

$$NTM[O(t(n))] \Leftrightarrow DTM[O(2^{O(t(n))})]$$

Example: The satisfiability problem

$SAT=\{\langle\Phi\rangle \mid \Phi \text{ is a satisfiable Boolean formula.}\}$

Non-Deterministic algorithm:

- Guess an assignment of the variables
- Check if this is a satisfying assignment

SAT={<Φ>|Φ is a satisfiable Boolean formula.}

Time for n variables:

- Guess an assignment of the variables $O(n)$

- Check if this is a satisfying assignment $O(n)$

Total time: $O(n)$

SAT={<Φ>|Φ is a satisfiable Boolean formula.}

$$L \in NP$$

The satisfiability problem is an NP - Problem

Observation:

$$P \subseteq NP$$

Deterministic Polynomial

Non-Deterministic Polynomial

Open Problem: $P = ?\, NP$

Example: Does the Satisfiability problem have a polynomial time deterministic algorithm?

WE DO NOT KNOW THE ANSWER

Cook-Levin Theorem

SAT $\in$ P iff P=NP

The next is an NTM that decides the HAMPATH PROBLEM in nondeterministic polynomial time.

HAMPATH PROBLEM

N1 = "On input<G, s, t>, where G is a directed graph with nodes s and t:

1. Write a list of m numbers, p_1, …, p_m, where m is the number of nodes in G. Each number in the list is nondeterministically selected to be between 1 and m.

HAMPATH PROBLEM

2. Check for repetitions in the list. If any are found, reject.

3. Check whether s = p_1 and t = p_m. If either fails, reject.

4. For each i between 1 and m-1, check whether (p_i, p_{i+1}) is an edge of G. If any are not, reject. Otherwise, all tests have been passed, so accept."

NP-completeness

A language B is NP-complete if it satisfies two conditions:

1. B is in NP, and

2. every A in NP is polynomial time reducible to B.

Examples for NP-completeness

1, SAT is NP-complete.

2, CLIQUE is NP-complete.

3, VERTEX-COVER is NP-complete.

4, HAMPATH is NP-complete.

5, UHAMPATH is NP-complete.

6, SUBSET-SUM is NP-complete.

- CLIQUE={<G, k>|G is an undirected graph with a k-clique}

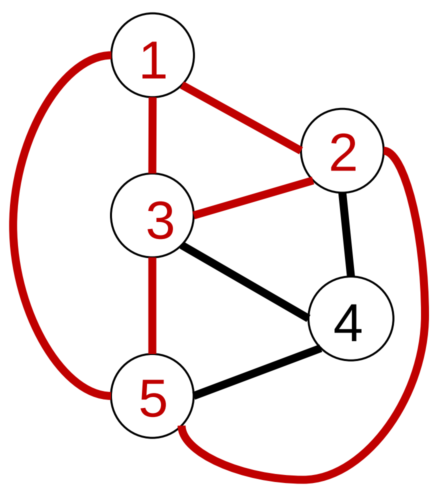
- SUBSET-SUM={<S, t>|S={$x_1,\ldots,x_k$} and for some {$y_1,\ldots,y_l$} $\subseteq$ {$x_1,\ldots,x_k$}, we have $\sum y_i = t$ }

CLIQUE

Given an undirected graph G=(V,E) and an integer $K \geq 2,$ is there a subset C of V with $|C| \geq K$

such that for all $u_i, v_j \in C$ there is an edge between u_i and v_j

Examples:



G

$(G,3)?$

$(G,4)?$

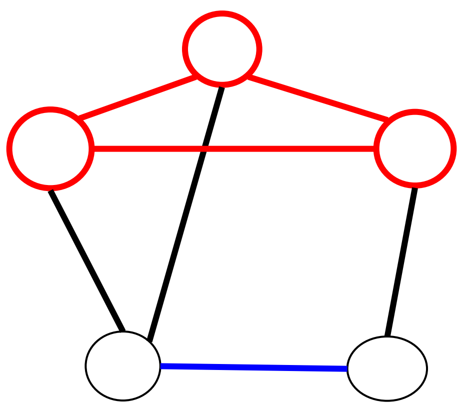
$(G,5)?$

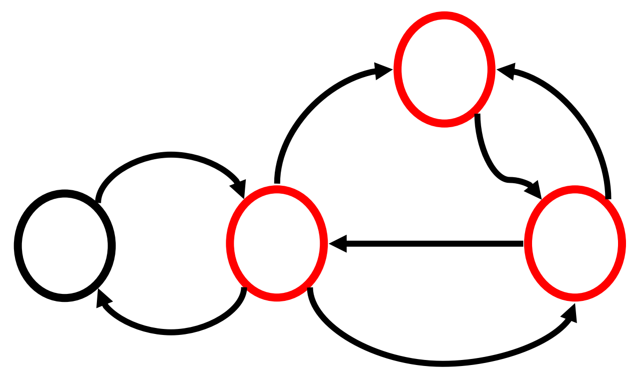
VERTEX-COVER

Given an undirected graph G=(V,E) and an integer $B \geq 2$, is there a subset C of V with $|C| \geq B$

such that C touches all edges of G?

Examples:





SUBSET SUM

- A set $Q = \{a_1, a_2, \ldots, a_n\}$ of positive integers and a postive integer d.
- Is there a subset of Q that adds up to d? That is,
  - Is there $S \subseteq Q$ such that $\sum_{a \in S} a = d$?

Proving a Problem in NP

Two steps:

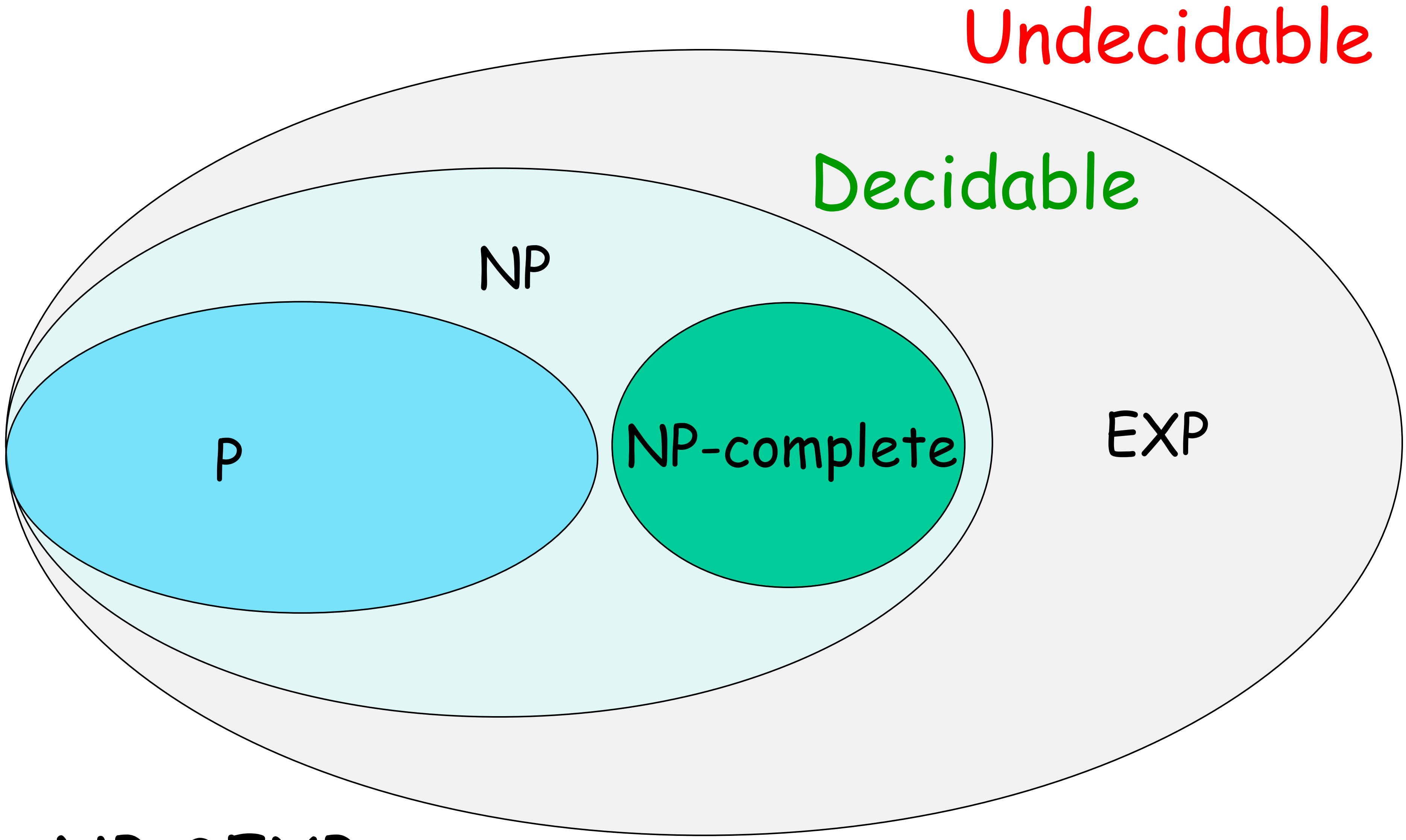
- Nondeterministically guess an answer in poly-time
- Deterministically verify the guessed answer is indeed an answer in poly-time

The Class EXP

Definition

The class EXP is the set of all problems that are decidable by exponentially-bounded TM's.

Hierarchy and an open question



NP=?EXP

NP-hard problems

- If all problems in NP can be polynomially reduced to a problem B, then B is called NP-hard.

- The class of NP-hard problem itself is not necessarily in NP.

NP-hard problems (Cont.)

- Informally NP-hard problems are at least as hard as or harder any problem in NP.
- if we can find an algorithm *A* that solves one of these NP-hard problems in polynomial time then we can construct a polynomial time algorithm for every problem in NP.

Example for NP-hard

- SUBSET-SUM is an NP-hard problem.
  - For example: given a set of integers, does any non empty subset of them add up to zero?
  - That happens to be NP-complete.

- The halting problem is NP-hard.
  - Which is : given a program and its input, will it run forever?
  - It has been proved that the halting problem is NP-hard but not NP-complete.

Example for NP-hard (Cont.)

- "is there a Hamiltonian cycle with length less than k" is NP-complete
  - it is easy to determine if a proposed certificate has length less than k.
- The optimization problem, "what is the shortest tour?", is NP-hard, since there is no easy way to determine if a certificate is the shortest.

Verifiers

Definition

- A decider machine V is called a verifier for a language L if

L = {w|V accepts <w, c> for some string c}

- The string c is called a certificate (or witness) for w

Certificates

- Every yes-instance of those problems has a short and easily checkable certificate
- Satisfiability — a satisfying assignment
- Hamiltonian Circuit — a Hamiltonian circuit
- In all cases one can easily prove a positive answer

verifier of polynomial time

- A verifier is said to be polynomial time if
  - it is a polynomial time Turing Machine, and
  - there is a polynomial $t(n)$ such that, for any w in L, there is a certificate c with $|c|$ less than or equal to $t(|w|)$

The alternative definition for Class NP

Definition

- The class of languages that have polynomial time verifiers is called the NP class
- i.e. all problems from NP class have a polynomial time verifier.

Equivalence

Theorem

The two definitions of NP are equivalent.

Proof

If $L \in NTIME[n^k]$, then there is an NTM such that $x \in L$ if and only if there is an accepting computation path in NTM(x). Furthermore, the length of these paths is in $O(|x|^k)$.

Proof (Cont.)

Using (some encoding of) these computation paths as the certificates.

We can construct a polynomial time verifier for L which simply checks that each step of the computation path is valid.

Proof (Cont.)

- Conversely, if L has a polynomial-time verifier V, then we can construct a NTM that first "guesses" the value of the certificate by making a series of non-deterministic choices, and then simulates V with that certificate.
- Since the length of the certificate is polynomial in the length of the input, this machine is a nondeterministic polynomial-time decision procedure for L. (QED)

Space complexity

$SPACE(f(n)) = \{ L \mid L$ is a language decided by a $O(f(n))$ space deterministic Turing machine}

$NSPACE(f(n)) = \{ L \mid L$ is a language decided by a $O(f(n))$ space nondeterministic Turing machine}

Example: SAT

M_1="On input 〈Φ〉 ,where Φ is a Boolean formula:

1. For each truth assignment to the variables $x_1, x_2, \ldots x_n$ of Φ:
2. Evaluate Φ on that truth assignment.
3. If Φ ever evaluated to 1, *accept*; if not, *reject*."

Space complexity O(n)

PSPACE

PSPACE is the class of languages that are decidable in polynomial space on a deterministic Turing machine

$$PSPACE = \bigcup_{k} SPACE\left(n^{k}\right)$$

Savitch's theorem

Savitch's theorem

$NSPACE(f(n)) \subseteq SPACE(f^2(n))$

The relationship among the complexity classes defined so far

$P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$

Thank You