

Algorithms & Complexity

Co-NP. EXPTIME and NEXPTIME.

Anton Bryl - abryl@computing.dcu.ie

CA313@Dublin City University. 2009-2010.

November 16, 2009

Hierarchy of problems

- ▶ We know that P is included in NP (a deterministic Turing machine is a particular example of a non-deterministic Turing machine).
- ▶ It is thought that $P \neq NP$, but it hasn't been proved yet.
 - ▶ For many NP problems no polynomial-time algorithms have been found but it has not been proved that no one exist.
 - ▶ If a polynomial algorithm to solve a NP -complete problem would be found, this would prove that $P = NP$ (all problems in NP would reduce to this problem belonging to P).
 - ▶ If $P=NP$, all problems in P are NP -complete, except those that have only positive instances or only negative instances.
- ▶ some np -hard problems are not in NP . Example: the halting problem (it is undecidable).

NP -intermediate Problems

- ▶ If $P \neq NP$, then are there problems which are neither in P nor NP – *complete*?

NP -intermediate Problems

- ▶ If $P \neq NP$, then are there problems which are neither in P nor NP – *complete*?
- ▶ The answer is yes (Ladner's theorem). These problems are called NP -intermediate (class NPI).
- ▶ There are real problems in NP for which neither polynomial-time algorithms nor NP -completeness proofs have been found. It makes sense to assume that these belong to NPI , until we know differently.

NP-intermediate Problems

Examples

- ▶ Graph isomorphism
 - ▶ An isomorphism of graphs G and H is a bijection between the vertex sets of G and H : $f : V(G) \rightarrow V(H)$ such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H . Thus two graphs are isomorphic if one can be transformed into the other simply by renaming vertices. Graph isomorphism (is graph G_1 isomorphic to graph G_2 ?) is in NP and is suspected to be neither in P nor NP-complete.
- ▶ Composite number (Given $N \in \mathbb{Z}_+$, find $m, n \in \mathbb{Z}_+$ such that $mn = N$)

NP-intermediate Problems

Examples: things sometimes change...

- ▶ Linear programming was considered *NP*-intermediate but in 1979 it was proven to be in *P*
 - ▶ Linear programming is related to the problem of solving a system of linear inequalities.
 - ▶ Problem was first shown to be solvable in polynomial time by Leonid Khachiyan in 1979. In 1984 Narendra Karmarkar found an algorithm solving this problem in polynomial time (Karmarkar's algorithm).

Co-NP Problems

Definition (Complement)

The *complement* of a decision problem X , denoted \bar{X} , is the decision problem in which the “Yes” instances of \bar{X} are the “No” instances of X and vice versa.

Examples

- ▶ Consider the problem: “is a number a prime number?”. Its complement is to determine whether a number is a composite number (a number which is not prime).
- ▶ Consider the subset sum problem: Given a set of integers $S = \{i_1, i_2, \dots, i_n\}$, does any (non-empty) subset $A \subseteq S$ sum to 0?

Its complement asks: Given a finite set of integers does every non-empty subset have a nonzero sum?

Co- NP Problems

Definition (Co- NP)

Co- NP is the class of complements of NP problems.

Example

- ▶ The subset sum problem is in NP (actually it is NP -complete) thus its complement problem is in co- NP .

Hierarchy of problems

▶ $P = co-P?$

Hierarchy of problems

- ▶ $P = co-P$?
 - ▶ Yes. If a problem X is in P , it can be solved by a deterministic Turing machine M in polynomial time. The same Turing machine with reversed accepting and rejecting states can be used to solve the complementary problem.

Hierarchy of problems

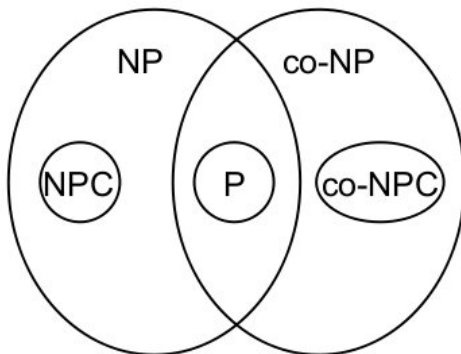
- ▶ $P = co-P$?
 - ▶ Yes. If a problem X is in P , it can be solved by a deterministic Turing machine M in polynomial time. The same Turing machine with reversed accepting and rejecting states can be used to solve the complementary problem.
- ▶ $NP = co-NP$?

Hierarchy of problems

- ▶ $P = co-P$?
 - ▶ Yes. If a problem X is in P , it can be solved by a deterministic Turing machine M in polynomial time. The same Turing machine with reversed accepting and rejecting states can be used to solve the complementary problem.
- ▶ $NP = co-NP$?
 - ▶ Probably no. To prove that the problem answer is “yes”, you just need to check that *one* given solution that yields a “yes”. However to prove a negative answer you must check that *all* possible solutions fail.

Hierarchy of problems

Assuming $P \neq NP$ and $NP \neq co - NP$:



Hierarchy of problems: additional remarks

- ▶ If $NP \neq co-NP$ then $P \neq NP$: the class P is equal to its complement.
- ▶ if there is an NP-complete problem L with complementary \bar{L} in NP , then $NP = co-NP$.
 - ▶ Consider a problem $Q \in NP$. You can reduce Q to L .
 - ▶ This reduction can be used also to reduce \bar{Q} to \bar{L} (instances are the same).
 - ▶ Thus if $\bar{L} \in NP$, $\bar{Q} \in NP$ and $co-NP \subseteq NP$.
 - ▶ $\bar{Q} \in NP \Rightarrow \bar{\bar{Q}} = Q \in co-NP$. Thus $NP \subseteq co-NP$.

Class EXPTIME

- ▶ EXPTIME is the class of all decision problems solvable by a deterministic Turing machine in $O(2^{p(n)})$, where $p(n)$ is any polynomial function.



$$\text{EXPTIME} = \bigcup_{k \in \mathbb{N}} \text{DTIME} \left(2^{n^k} \right)$$

- ▶ A problem is EXPTIME-complete if it is in EXPTIME and all other problems in EXPTIME has a polynomial-time reduction to it.

Class NEXPTIME

- ▶ NEXPTIME: all the same, but with a non-deterministic Turing machine.

Class NEXPTIME

- ▶ NEXPTIME: all the same, but with a non-deterministic Turing machine.
- ▶ NEXPTIME is the class of all decision problems solvable by a non-deterministic Turing machine in $O(2^{p(n)})$, where $p(n)$ is any polynomial function.



$$\text{EXPTIME} = \bigcup_{k \in \mathbb{N}} \text{NTIME} \left(2^{n^k} \right)$$

- ▶ A problem is NEXPTIME-complete if it is in NEXPTIME and all other problems in NEXPTIME has a polynomial-time reduction to it.

Hierarchy of problems

- ▶ $NP \subseteq EXPTIME$
- ▶ $EXPTIME \subseteq NEXPTIME$

Hierarchy of problems

- ▶ $NP \subseteq EXPTIME$
- ▶ $EXPTIME \subseteq NEXPTIME$
- ▶ $EXPTIME = NEXPTIME?$

Hierarchy of problems

- ▶ $NP \subseteq EXPTIME$
- ▶ $EXPTIME \subseteq NEXPTIME$
- ▶ $EXPTIME = NEXPTIME?$
 - ▶ Unknown; commonly thought that no.

Hierarchy of problems

- ▶ $NP \subseteq EXPTIME$
- ▶ $EXPTIME \subseteq NEXPTIME$
- ▶ $EXPTIME = NEXPTIME?$
 - ▶ Unknown; commonly thought that no.
 - ▶ But it is known that if $P = NP$, then $EXPTIME = NEXPTIME$.

Hierarchy of problems

