The Class P

Def. P is the class of languages decidable on a determinstic, single tape TM in time bounded by some polynomial (in size of input)

P contains all problems that are realistically solvable

P turns out to be invariant over many models of computation

May still have problems in P we don't think are realistically solvable (e.g. $O(n^{v})$) where **v** *is very large no.)*

Gives us fundamental way of grouping problems:

P **not in P**

1

The Class NP

Many problems have as their only solution brute-force search

May not be solvable in polynomial time

Def. NP is the class of languages decidable on a nondeterminstic, single tape TM in time bounded by some polynomial (in size of input)

Ex. Travelling Salesman problem: **solution < k miles**

Only known solution is to try all tours- exponential time for deterministic TM

Nondeterminstic TM can guess the tour, then check if its length is less than k in polynomial time!

2

Problems in P

PATH = {<G,s,t> | G is dir. graph & has path s to t}

Th. Path is in P

Proof: Use breadth-first search of nodes in G reachable from s.
On input <G,s,t>:

1. **Mark node s.**
2. **Repeat until no new nodes marked:**
   Scan all edges of G. If any edge (a,b) with a marked and b unmarked, then mark b.
3. **If t is marked, accept; ow reject.**

Time: 1: constant time 3: constant time
2: at most O(|N|), N the nodes in G

More Problems in P

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

Relatively prime iff 1 is the largest common int. divisor
Ex. 10 and 21 are relatively prime, though not prime!

Note: if x and y stored in binary, then searching through all possible integer divisors will be exponential in length of <x,y>!!

What can yield polynomial time algorithm?
Euclidean algorithm to compute GCD

Every CFL

Use Chomsky Normal form? Derivation limited to 2n-1
But there are exponentially many such derivations!
Can use dynamic programming.

Usually must avoid brute force searches to show in P

Problems in NP

A Hamiltonian path is one that goes through each node in the graph exactly once.

HamPATH = {<G,s,t> | G is dir. graph & has Hamiltonian path s to t}

Idea: Guess path starting with s nondeterministically, verifying that never repeat a node, and include all nodes.

Given a path, easy to check if its Hamiltonian?

Typically, easy to verify an instance; hard to verify existence.

Alt. Def. of NP: Class of languages which have a determinstic, polynomial time verifier.

More Problems in NP

Graph Coloring: G = (V,E), integer k > 2.
Are the vertices of G colorable with k colors, so that adjacent nodes (with edge between) are colored with different colors?

(In P when k = 1,2,3)
Related to register allocation problem in compilers

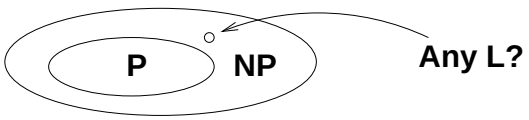
Traveling Salesman

Satisfiability of Boolean formulas

Graph partitioning

Processor Scheduling

Does P = NP???



Big open problem! No one has been able to show....

General belief: $P \neq NP$

Best known methods for solving problems in NP are determinstically exponential.

Def: A language B is *NP-complete* **if**

1. **B is in NP**
2. **Every A in NP is reducible to B in poly. time**

If could show an NP-complete language has a det. poly. time algorithm, then we'd have P = NP.

Problem SAT

Boolean variables (values 0, 1 for false, true)
Boolean operations AND OR NOT
^ v $\overline{}$

Boolean formula: well-formed expression of Boolean variables, operations.

Ex. $(\overline{x} \wedge y) \vee (x \wedge \overline{z})$

Def: A Boolean formula is *satisfiable* **if there is some assignment of 0's and 1's that makes the formula evaluate to 1.**

Ex. x = 0, y = 1, z = 1

SAT = { F | F is a satisfiable Boolean formula }

TH: SAT is NP-complete

SAT is in NP:

Proof: A nondet. poly. time TM can guess an assignment to a formula F, and accept if the assignment satisfies F.

Every A in NP is reducible to SAT in poly. time:

Proof idea: Use formula $F_{A,w}$ *to describe the computation of a polynomial time TM accepting w in A, with formula* $F_{A,w}$ *satisfiable* $\longleftrightarrow$ *w in A.*

Other NP-complete Problems

Hamiltonian Path

Graph Coloring

Graph partitioning

Traveling Salesman

2 Processor Scheduling

See Garey & Johnson, ''Computers and Intractability: A Guide to the Theory of NP-Completeness'', W.H. Freeman & Co.

Th. If B is NP-complete and B is in P, then P = NP.

Practical Consequences of NP-Completeness

If show problem in NP....don't bother searching for polynomial time algorithm!

Many solutions start by showing problem is NP-complete.

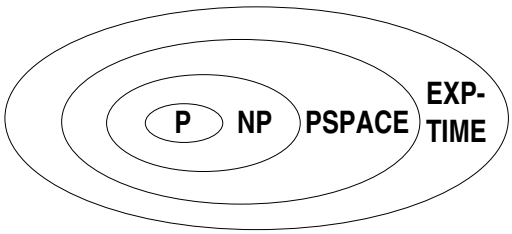
If need more efficient algorithm than brute force search (exponential time) can use heuristic methods

May not give the best solution eg minimal coloring

Even though your first version of practical problem may be NP-complete, you may be able to change formalization of problem to get new problem in P

MANY interesting problems are NP-complete!

Other Complexity Classes



PSPACE: languages accepted by some TM which uses only poly. no. tape cells as function of input. *(Det. same as Nondet. Polyspace--WHY?)*

EXPTIME: languages accepted by some TM in no. of steps exponential in the size of input.

(Det. NOT same as Nondet.--WHY?)

Can prove that P $\neq$ EXPTIME