CS3172: Advanced Algorithms
Introduction to Complexity - II

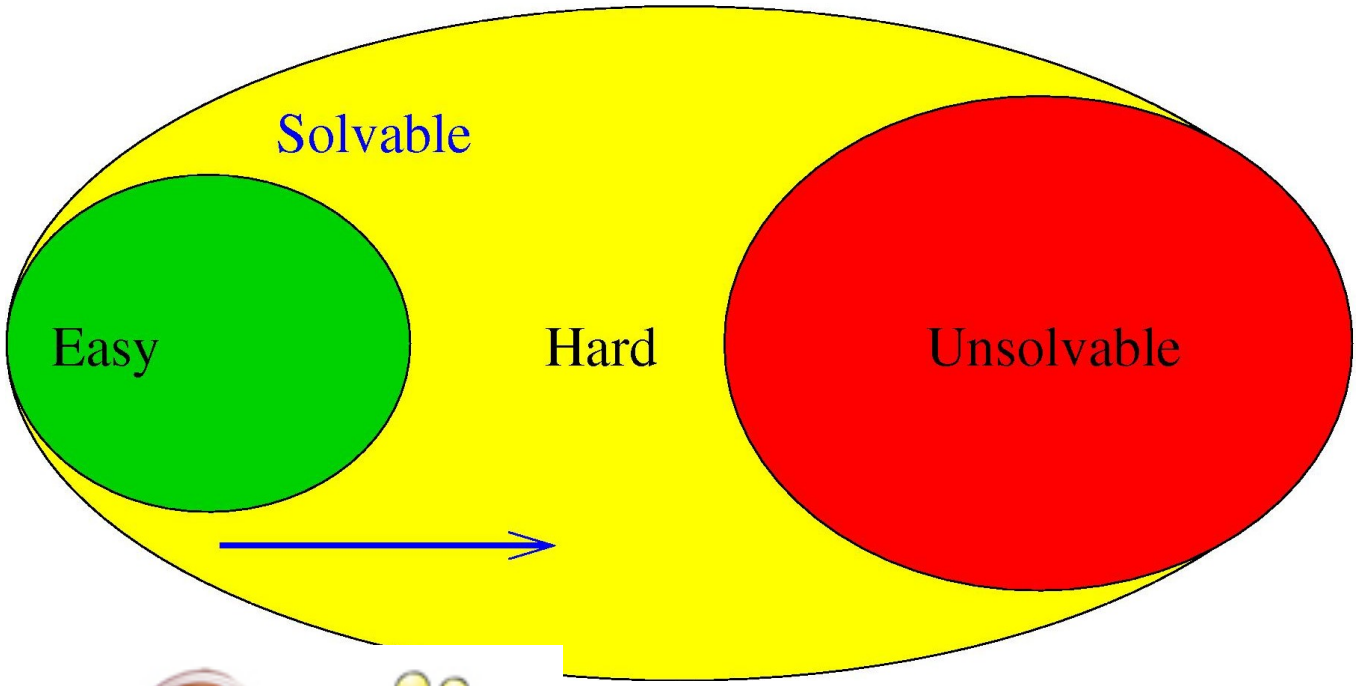
Howard Barringer

Room KB2.20: email: howard.barringer@manchester.ac.uk

March 2006

Introduction to Complexity - II

Classifying Problems by their Difficulty



Bin Packing

Given bins, each of capacity C, and n objects of sizes $s_1, s_2, \ldots, s_n$ such that $0 \leq s_i \leq C$ for all i, $0 \leq i \leq n$.

Decision Problem:
For a given number k, can the n objects be packed into k bins?

Knapsack

Given a knapsack of capacity C (> 0) and n objects of (positive whole number) sizes $s_1, s_2, \ldots, s_n$ with respective (positive whole number) costs $c_1, c_2, \ldots, c_n$.

Decision Problem:
Given a positive integer k, is there a subset of the objects that fits in the knapsack and has total cost k?

Subset Sum

Even simpler variant of knapsack ...

Given a capacity C and objects of (positive integer) size $s_1, s_2, \ldots, s_n$

Decision Problem:
Is there a subset of the given objects whose sizes add to match exactly the capacity C?

Graph Colouring

Given a graph $G = (V, E)$, a colouring of G is a mapping

$$C : V \to S$$

where S is a finite set (of colours) such that

$$C(v) \neq C(w) \textbf{ if } (v, w) \in E$$

Decision Problem:

Give[illegible] ber k, is there a colouring of G using

Hamiltonian Paths/Cycles

Given a graph $G = (V, E)$, a Hamiltonian path (resp. cycle) is a path (resp. cycle) that passes through every vertex exactly once.

Decision Problem:
Does a given graph have a Hamiltonian path (or cycle)?

Travelling Salesman

A weighted variant of Hamiltonian paths ...

Given a weighted graph $G = (V, E)$ where each edge $e_i \in E$ has cost c_i.

Decision Problem:
For a given total cost k, does the given graph have a Hamiltonian path with total cost at most k?

CNF Satisfiability

A literal l is either an atomic proposition or its negation.
A propositional formula is in Conjunctive Normal Form (CNF) if it is of the form:

$$\bigwedge_{i} \bigvee_{j \in J_i} l_{ij}$$

for example:

$$(p \vee q \vee r) \wedge (\neg p \vee \neg q) \wedge (p \vee \neg r)$$

Dec

Giv

Comment ...

The above small sample of decision problems (often presented as optimisation problems) appear rather straightforward.

Instances (and variants) of such problems have numerous applications, from transport logistics and scheduling to theorem proving.

There are, however, no known easy solutions, i.e. have polynomially time bounded algorithms that solve the problem.

Ver

A Guessing Game

Consider the subset sum problem given above.

Suppose one makes a guess at a solution, i.e. guesses a particular subset of objects.

It is very easy to check whether the proposed solution is indeed a correct solution. Simply sum the individual sizes of the objects and test whether the total is indeed equal to the desired capacity C.

Each of the above problems can be *solved* in this way. In particular, for each, the check is easy in our technical sense, i.e. the algorithm for (polynomially time-bounded.

Non-deterministic Algorithms for Decision Problems

These can be characterised as having two phases.

- The non-deterministic phase.
  An arbitrary string of symbols is written into memory as a proposed solution. Note, it could be complete gibberish - it doesn't matter!
- The deterministic phase.
  A deterministic algorithm is executed to determine whether the proposed solution is indeed correct. It terminates with answer yes, if so, and no, if it is not.

The  m is the sum of the costs of each

Non-deterministic Polynomial Time-bounded Algorithms

A non-deterministic algorithm is polynomially time-bounded if there is a polynomial function $f(n)$ such that for a given problem input of size n which as a yes answer, there exists an execution of the algorithm that yields the answer yes in at most $f(n)$ steps.

The Complexity Class *NP*

We define the class *NP* as the set of all (decision) problems that can be solved using a non-deterministic polynomially time-bounded algorithm.

Bin-packing, Knapsack, Subset sum, Graph colouring, Hamiltonian paths/cycles, Travelling Salesman, CNF-satisfiability (or SAT) are all in the class *NP*.

Relation between P and NP

Theorem:

$$P \subseteq NP$$

Proof:

Each (decision) problem in P has a deterministic polynomial time-bounded algorithm. A non-deterministic polynomial algorithm can thus be constructed for each member of $P$by using the deterministic algorithm as the checking phase and ignoring completely the guessing phase, i.e. do 0 steps.

Beliefs ...

It is unknown whether

- $P = NP$
- $P \subset NP$

However, it is believed that NP is indeed much bigger than P, but for no problem established to be in NP has it been proved that the problem is not in P

Characterising the hardest problems in *NP*

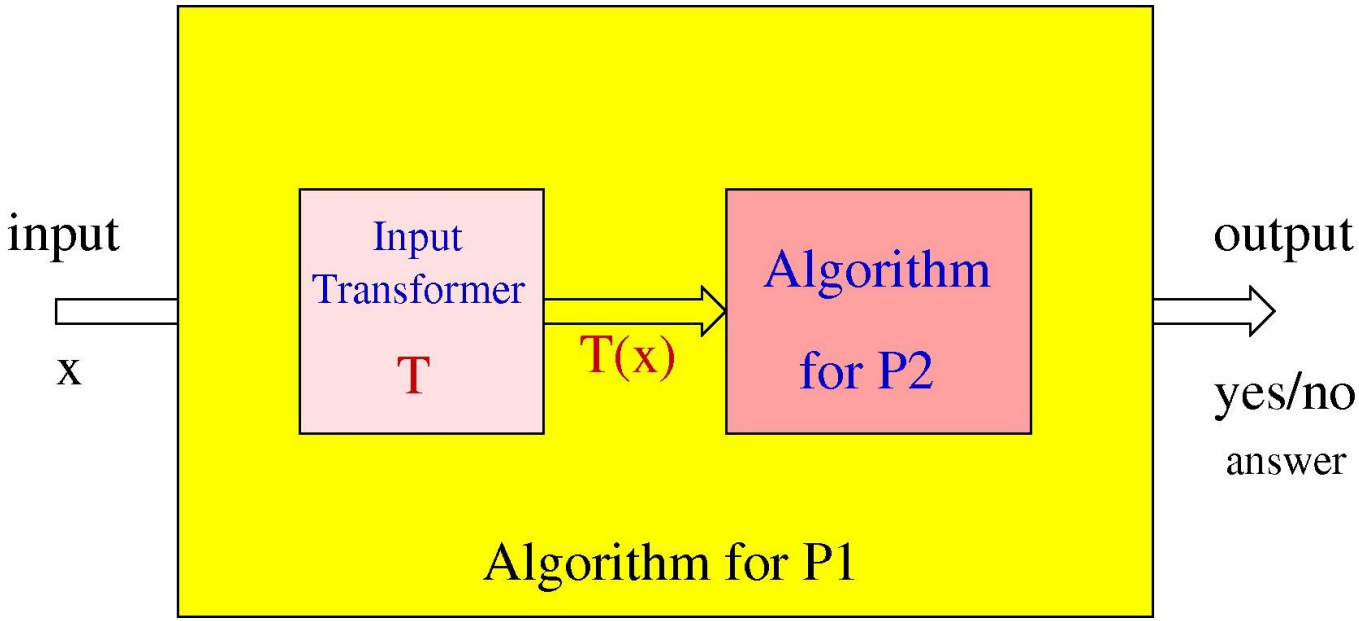
The above set of problems certainly vary in their deterministic complexity, e.g.

- Travelling salesman is $O(n!)$
- CNF-satisfiability is $O(2^n)$

However, in terms of non-deterministic algorithms, amazingly they are all equally hard.

Indeed, if one finds a deterministic polynomial time algorithm to solve just one of the problems, then all of them will be solvable in deterministic polynomial time.

Reducing one problem to another



To s rm the input for P1 into input for

More formally ...

Let T be a function from the input set for (decision) problem Π_1 to the input set for (decision) problem Π_2. T is a polynomial reduction from Π_1 to Π_2 if

1. T can be computed in (deterministic) polynomial-bounded time
2. For every input x for Π_1, the correct answer for Π_2 on $T(x)$ is the same as the correct answer for Π_1 on x.

We say that Π_1 is polynomially reducible to Π_2 if there exists a poly $_1$ to Π_2.

Reductions in P

Theorem:

If Π_1 is polynomially reducible to Π_2 and Π_2 is in P, then Π_1 is in P.

Proof: Straightforward. The sum and product of two polynomials functions are still polynomial.

NP-Complete complexity class

A decision problem Π is said to be *NP*-complete if it is in *NP* and for every other problem Π' in *NP*, then Π' is polynomially reducible to Π.

Theorem:

If any *NP*-complete problem is in *P*, then $P = NP$.

Theorem: due to Stephen Cook (1971)

The  m is *NP*-complete.

A Final Word on Space

We define the complexity class $DSPACE(f(n))$ to contain those problems whose deterministic algorithms solve in space bounded by $f(n)$

Similarly, $NSPACE(f(n))$ are those problems whose nondeterministic algorithms solve in space bounded by $f(n)$

Then,

$$PSPACE = \bigcup_{i \geq 1} DSPACE(n^i)$$

$$NSPACE = \bigcup_{i \geq 1} NSPACE(n^i)$$

$PSPACE$ is a huge class, probably way above P and NP

$DSPACE(\log n$ is a small class, within P, and very small within $PSPACE$.

Inte  ıat

A small bit of the Complexity Hierarchy

Theorem:

$$DSPACE(\log n) \subseteq P \subseteq NP \subseteq PSPACE$$

where at least one of the containments is strict, but it is unknown which it is.

Not unsurprisingly, there are many problems beyond *PSPACE*. *EXPSPACE*, for example, is the class $\bigcup_{i \geq 1} DSPACE(2^{n^i})$. Recognizing whether two regular expressions (using union, con  nd squaring) denote different