Computability Theory

All languages

languages that we can describe

languages that we can decide
The map-colouring problem

Is it possible to paint a colour on each region of a map so that no neighbours are of the same colour?

Not all decidable problems are equally hard to solve!

Yes! If you can use as many colours as you like.
The map-colouring problem

Is it possible to paint a colour on each region of a map so that no neighbours are of the same colour?

Rarely possible with only two colours!

But we find this out quickly.

Somewhat easier with three colours, though not always possible.

But can take long time to decide.
The map-colouring problem

Is it possible to paint a colour on each region of a map so that no neighbours are of the same colour?

Much easier with four colours!

But is it always possible?

K-colouring of maps (planar graphs)

Question: Can we colour any given map with at most $K$ colours?

- $K=1$, Easy to decide. Only maps with zero or one region are 1-colourable.
- $K=2$, Easy to decide. Impossible as soon as 3 regions touch each other.
- $K=3$, No known efficient algorithm to decide. However it is easy to verify a solution.
- $K \geq 4$, All maps are $K$-colourable (hard proof). Does not imply it is easy to find a $K$-colouring for a given map.

All these are decidable problems. But of different difficulty.
3-colouring of maps

- Seems hard to solve in general.
  - If I give you a map, it's hard to find a correct colouring, or be able to tell that no colouring is possible.

- It's easy to verify when a solution is given.
  - If I give you a map and a colouring, it's easy for you to check whether the colouring is correct (o.e. no adjacent cell has same colour.)

- This is a special type of problem, called NP-complete.
  - Hard to solve in general, but easy to verify whether a given solution is correct.

NP-complete problems

- NP = Nondeterministic Polynomial Time

- Many practical problems are NP-complete.
  - Some books list hundreds of such problems.

- If any of them is easy, they are all easy. (Remember what we said about reducing a problem to another when we discussed decidability.)

- In practice, some of them may be solved efficiently in some special cases.
Examples of NP-complete problems

- **Boolean satisfiability**: Given a Boolean expressions, is there an assignment of the boolean variables making the formula evaluate to true?

- **3-SAT**: Similar to Boolean satisfiability, but the expression must contain only 3-variable sub-expressions, separated by AND statements.

- **Travelling Salesman**: Given a set of cities and distances between them, what is the shortest route to visit each city once.

- **KnapSack**: Given items with various weights, is there of subset of them of total weight K.

Computability Theory

- **All languages**
- **Languages we can describe**
- **Languages we can decide**
- **NP-complete problems**
Boolean satisfiability example

• Question: Is there an assignment of the boolean input variables {A, B, C, D} that makes the output E evaluate to true?

\[ E = (A \lor (\neg B) \lor (\neg C)) \land (A \lor B \lor D) \]

• Complexity of Boolean satisfiability:
  – How many operations to find a solution?
  – How many operations to verify a given solution?
  – How does this change as a function of the number of input variables?

Reduction

How do we know if a new problem is NP-complete?

Reduce a known NP-complete problem to this new problem.
Reduction from 3-SAT to graph colouring

- Construct a graph that will be 3-colorable if and only if the 3-SAT instance is satisfiable.

- Graph construction:
  - Create 3 special nodes T, F, X, joined in a triangle. (Note: T=True, F=False, X=other)
  - For each variable in the SAT problem, create 2 nodes $x_i$ and NOT($x_i$), connected by an edge. Connect all of these nodes to X.
  - Add 5 nodes and 10 edges to represent each 3-variable sub-clause.

- Colouring rules:
  - Each $x_i$, NOT($x_i$) pair must be a different colour.
  - Each must be a different colour than X.
  - X, T, and F must get different colors.

- Any 3-colouring of this graph gives a valid solution to the boolean expression.
Tractable Problems (P)

- Fortunately, many practical problems are tractable.

- The name P stands for Polynomial- Time computable.
  - Polynomial = constant, linear, quadratic, … But not exponential!

- Computer scientists spend most of their time finding efficient solutions to tractable problems.
  - But also lots of useful work in figuring out how to get approximate solutions for non-tractable (i.e. NP-complete) problems.

Examples of Tractable Problems (P)

- 2-colorability of maps.
- Primality testing.
- Solving NxNxN Rubik’s cube.
- Finding a word in a dictionary.
- Sorting elements in a list.

These are “easy” problems!
Complexity of tractable problems

- Polynomial time solution
- Quadratic time
- Linear time

Complexity Theory

- Decidable languages
- NP-complete
- NP

- P = NP?

Note: NP-complete problems are those problems which are the hardest in the NP class.
P=NP?

- THE fundamental question of theoretical computer science:
  For all problems for which a computer can verify a given solution quickly (i.e. in polynomial time), can the computer also find that solution quickly.

- This is the million dollar question! (Quite literally…)
  http://www.claymath.org/millennium/P_vs_NP/

- In 2002, 100 researchers were asked “Do you think P=NP?”
  - 61 believed the answer is no,
  - 9 believed the answer is yes
  - 22 were unsure
  - 8 believed the question may be impossible to prove or disprove.

Some progress?

P ≠ NP

Vinay Deolalikar
HP Research Labs, Palo Alto
vinay.deolalikar@hp.com
August 6, 2010
Approximating NP-hard problems

- **Boolean satisfiability:** given a boolean expression, is there an assignment of the boolean variables making the formula evaluate to true?

- Small expressions can be solved by searching through all assignments.

- Large expressions can (sometimes) be solved by random search. E.g. Genetic algorithms.

Beyond NP-completeness

- **PSPACE Completeness:**
  Problems that require a polynomial (P) amount of memory (i.e. space) to be solved.

- There are many such problems.
- We currently don’t know if $P = PSPACE$.

- Many other complexity classes exist (in addition to $P$, $NP$, $PSPACE$), e.g. $EXPTIME$, $EXPSPACE$, $Co-NP$, $PP$, etc.
PSPACE completeness

- **Geography Game:**
  Given a set of country names:
  Angola, Canada, Cuba, France, Italy, Japan, Korea, Vietnam

- A two player game:
  One player chooses a name. The other player must choose a name that starts with the last letter of the previous name and so on. A player wins when his opponent cannot play any name.

---

Generalized Geography

- Given an arbitrary set of names: \(w_1, \ldots, w_n\).

- Is there a winning strategy for the first player to the previous game?

- Can answer this by storing only a list of all the names.
  - So memory (space) requirement of the algorithm is polynomial in the size of the problem (n).
Complexity Theory

NP = P-space?

Theoretical Computer Science

- Challenges of theoretical computer science:
  - FIND efficient solutions to many problems.
  - PROVE that certain problems are NOT computable within a certain time or space.

- Consider new models of computation.
  (Such as a Quantum Computing – more on this in a few weeks.)
Take-home message

• Know the difference between problems that are P (known to be “easy”) and those that are NP (possibly “hard”).

• Be able to name some examples of NP-complete problems.

• Be able to name some examples of tractable (polynomial-time) problems.

• Understand the idea of reduction, as used in complexity theory.