CSE 417: Algorithms and Computational Complexity

Winter 2002

Justin Campbell

1

Russell's Paradox

- Similar in flavor to the Halting problem.
- Consider the set of all sets that don't contain themselves.
- Example: { a, b, {a} }
- Does this set contain itself?

2

Reductions

- We write: $L \leq R$
- We transform an instance of L into an instance of R such that R's answer is L's.
  - Function L(x)
    - Run program T to translate input x for L into an input y for R
    - Call a subroutine for problem R on input y
    - Output the answer produced by R(y)

3

Reductions

- If $L \leq^p R$ and R is efficiently solvable then so is L. Using the contrapositive, if L is provably slow, then R must be.
- If L is $\Omega(P(n))$ and the reduction is T(n) then R is $\Omega(P(n) - T(n))$

4

Reductions Exercise

- Show: Vertex-Cover $\leq^p$ Independent Set
- Vertex-Cover:
  - Given an undirected graph G=(V,E) and an integer k is there a subset W of V of size at most k such that every edge of G has at least one endpoint in W? (i.e. W covers all vertices of G).
- Independent-Set:
  - Given a graph G=(V,E) and an integer k, is there a subset U of V with $|U| \geq k$ such that no two vertices in U are joined by an edge.

5

Properties of polynomial-time reductions

- Theorem: If $L \leq^p R$ and $R \leq^p S$ then $L \leq^p S$
- Proof idea:
  - Compose the reduction T from L to R with the reduction T' from R to S to get a new reduction T''(x)=T'(T(x)) from L to S.

6

Computational Complexity

- Classify problems according to the amount of computational resources used by the best algorithms that solve them
- Define:
  - $TIME(f(n))$ to be the set of all problems solved by algorithms having worst-case running time $O(f(n))$
  - Ex: Sorting is in $TIME(n\log n)$.

Polynomial time

- Define P (polynomial-time) to be
  - the set of all problems solvable by algorithms whose worst-case running time is bounded by some polynomial in the input size.
- $P = \bigcup_{k\geq 0} TIME(n^k)$

Polynomial versus exponential

- We'll say any algorithm whose run-time is
  - polynomial is good
  - bigger than polynomial is bad
- Note:
  - n^{100} is bigger than $(1.001)^n$ for most practical values of n but usually such run-times don't show up
  - There are algorithms that have run-times like $O(2^{n/22})$ and these may be useful for small input sizes.

Beyond P?

- There are many natural, practical problems for which we don't know any polynomial-time algorithms
- e.g. Vertex-Cover, Independent-Set
- e.g. Traveling Salesman Problem
- e.g. Satisfiability

Satisfiability

- Boolean variables $x_1,\ldots,x_n$
  - taking values in $\{0,1\}$. 0=false, 1=true
- Literals
  - x_i or $\neg x_i$ for $i=1,\ldots,n$
- Clause
  - a logical OR of one or more literals
  - e.g. $(x_1 \vee \neg x_3 \vee x_7 \vee x_{12})$
- CNF formula
  - a logical AND of a bunch of clauses

Satisfiability

- CNF formula example
  - $(x_1 \vee \neg x_3 \vee x_7 \vee x_{12}) \wedge (x_2 \vee \neg x_4 \vee x_7 \vee x_5)$
- If there is some assignment of 0's and 1's to the variables that makes it true then we say the formula is satisfiable
  - Is the following formula satisfiable?

    $x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge \neg x_3$
- Satisfiability: Given a CNF formula F, is it satisfiable?

Common property of these hard problems

- There is a special piece of information, a short hint or proof, that allows you to efficiently verify (in polynomial-time) that the answer is correct. This hint might be very hard to find.
- e.g.
  - Independent-Set, Clique: the set of vertices
  - Satisfiability: an assignment that makes F true.

13

The complexity class NP

- NP consists of all problems where one can verify the answers efficiently (in polynomial time) given a short (polynomial-size) hint.
- The only obvious algorithm for most of these problems is brute force:
  - try all possible hints and check each one to see if it works.
  - Exponential time.

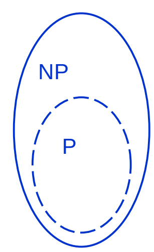
14

Unlike undecidability

- Nobody knows if all these problems in NP can all be done in polynomial time, i.e. does P=NP?
  - one of the most important open questions in all of science.
  - huge practical implications
- How are P and NP related?

15

P and NP



16

NP-hardness & NP-completeness

- Alternative approach
  - show that they are at least as hard as any problem in NP
- Rough definition:
  - A problem is NP-hard iff it is at least as hard as any problem in NP
  - A problem is NP-complete iff it is both
    - NP-hard
    - in NP

17

NP-hardness & NP-completeness

- Definition: A problem R is NP-hard iff every problem $L \in NP$ satisfies $L \le^p R$
- Definition: A problem R is NP-complete iff R is NP-hard and $R \in NP$
- Not obvious that such problems even exist!

18