Introduction To NP-Completeness

CS410/510CS Lecture 3

Bart Massey <bart@cs.pdx.edu>
PSU CS Department
January 25, 2001

Overview

- Computational Complexity
- NP-Completeness
- Applications

Order Analysis and Computational Complexity

- Motivation: estimate problem difficulty
- Two approximations: Consider only
  - *growth rate* of difficulty with *instance size*
  - *polynomial part* of growth rate
- Both approximations questionable
  - cryptography: difficulty of fixed-size instances
  - linear-time register allocation: huge constants

Two Theses

Polytime Thesis: Any realistic $O(n^k)$ problem is $O(n^3)$ problem

Church-Turing Thesis: The same problems are $O(n^k)$ problems on any computer (QP? Open)

Problem Descriptions

Need uniform notation for

- Distinguishing problem from class
- Distinguishing instance from problem
- Formulating size of instance

Problem Description Notation [Garey-Johnson]

Key Elements:

- Name: Identifies problem
- Instance: Lists all data comprising problem instance
- Question: Formulates yes/no "decision" question

Example:

EVEN SET

INSTANCE: A set S of integers.

QUESTION: Are all integers in S even?

Instance Size

Size of instance is minimum number of bits needed to represent instance.

What is size of EVEN SET instance?

What about EVEN ELEMENT?

Problems And Classes

Problem p is *in* class $\mathcal{C}$ when exists $\mathcal{C}$ algorithm for solving p instances.

p is *hard for* $\mathcal{C}$ when $\mathcal{C}$ algorithm for p also solves all instances in $\mathcal{C}$.

p is *complete for* $\mathcal{C}$ when p is in $\mathcal{C}$ and $\mathcal{C}$ hard.

The Class NP

Decision problem is in P if can answer yes/no in polytime.

Consider class of problems that can *check* yes answer in polytime.

- Same? No one knows
- Call this class NP

Nondeterministic Polynomial

Why NP? Because

- If you *guess* an answer (nondeterminism)
- You can *check* it in polytime

E.g. SAT, graph coloring, bin packing

$$P \stackrel{?}{=} NP$$

- Most think $P \neq NP$
- After many years, no proof
- Raises larger questions
- In this course, $P \neq NP$

Decision vs. Optimization Problems

- Many problems call for value, not decision
- Optimization problems call for best value
- Trick
  - Make optimization target part of instance
  - Binary search

Coclasses and Coproblems

- Note: NP decision problem is P check for 'yes' answer
- P check for 'no' answer?
  - These are co-NP problems
  - E.g. UNSAT, No-Coloring, No-Packing
  - Believed harder than NP
  - But $NP \stackrel{?}{=} co-NP$ open
  - Note $P = co-P$, so $P = NP$ would imply $NP = co-NP$

NP Complete

A problem is NP Complete if it is

- In NP: easy to check, but important
- Hard for NP: how to tell?

Many-One Reductions

If you have an NP-hard problem p, any problem that has “special case” p also NP-hard

E.g. GRAPH 5-COLORING NP-hard
implies GRAPH COLORING NP-hard

We say “GRAPH COLORING $\leq_m$ GRAPH 5-COLORING” and “GRAPH COLORING reduces to GRAPH 5-COLORING”

SAT Is NP Complete

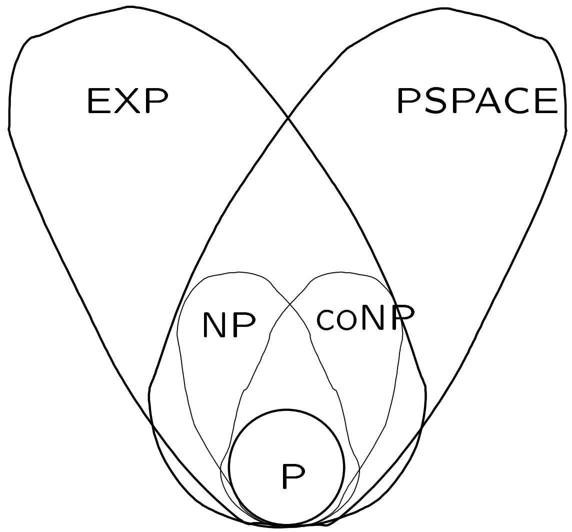
Can prove problems NP hard (thus NPC) by many-one reduction

But need base case

Cook's Theorem: SAT is NP-hard by definition

P, NP, and Beyond

Consider P, NP, co-NP. Can define more complex classes (w/ co-classes): EXP, PSPACE, oracle classes. Little known: mainly $P \subset EXP$



Reminder: Why We Care

Big excursion into theory. Why?

- Most comb. search problems in NP
- Should do one of
  - Prove your problem above NP
  - Prove your problem NPC
  - Give P algorithm for your problem

Techniques For NP Hardness Proof

- Simple reduction
- Polytime transformation + reduction
- Direct proof (almost never)

Instance Generalization

- Often given single instance: constant time!
- Still interested in instance "hardness"
- Generalize instance and find problem class
  - No "right" generalization
  - Answers wrong question
  - Still very useful

LOGS BOX STACKING

Given instance with

- 14 logs of given length
- 3-D box of given length

formulate

LOGS BOX STACKING

INSTANCE: n logs of length $l_1 \ldots l_n$ and unit width and height, k-dimensional box with sides $d_1 \ldots d_k$, such that $\prod_i d_i = \sum_j l_j$ and

$$\exists q \in \mathcal{R} \ . \ \forall i \ . \ l_i | q$$

QUESTION: Is there a packing of the logs into the box?

Proving LOGS BOX STACKING NPC

Two things to prove

- LOGS BOX STACKING in NP? Yes
- LOGS BOX STACKING NP-hard? Yes, by reduction from SUBSET SUM [G&J SP13]

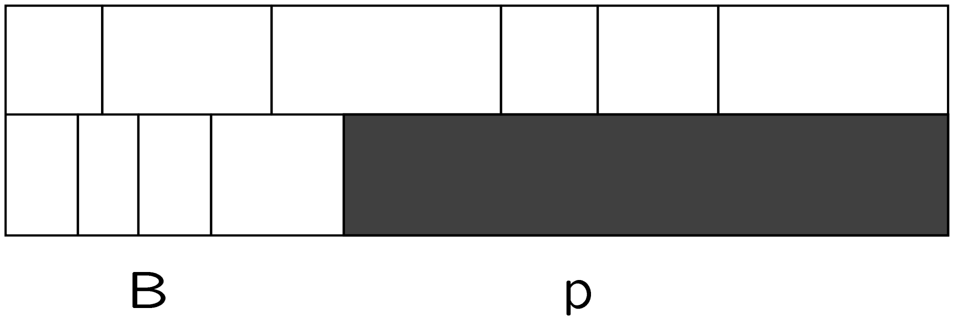
SUBSET SUM

INSTANCE: Finite set A, a size $s(a) \in \mathcal{Z}^+$ for each $a \in A$, positive integer B.

QUESTION: Is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = B$?

NP Hardness Proof For LOGS BOX STACKING

Consider SUBSET SUM instance with A, B. Make LOGS BOX STACKING instance with $2 \times k$ box and logs from A. Add spacer log p and choose k such that $\sum A + p = 2k$ and $B + p = k$ (or flip spacer).



Homework 1b

Homework 1b will ask you to generalize TENT PACKING and prove it NPC. I will give more hints with the actual assignment, shortly...