COMP3630/6360: Theory of Computation
Semester 1, 2022
The Australian National University

Time Complexity

This lecture covers Chapter 11 of HMU: Other Complexity Classes

- The Tautology Problem
- NP-Hardness and co-NP
- Optimisation Problems
- Other Complexity Classes

Additional Reading: Chapter 11 of HMU.

The Tautology Problem

The Tautology Problem

- To show that a problem is NP-complete, we need to show that it is in NP
- For some problems, we can show that they are NP-hard, but not that they are in NP
- NP-hardness of a problem implies that if this problem *were* in P, then P $=$ NP.

Definition 10.1

A boolean formula is a *tautology* if it evaluates to true for *all* truth value assignments. The *Tautology Problem* is the set of all boolean formulae that are tautologies.

The Tautology Problem

Is TAUT in NP?

- if it is, it is not obvious ...

The Complement of Taut is in NP

- that is, to decide if a formula is *not* a tautology
- guess variable assignment, accept if formula evaluates to false

Key Message

- The nondeterministic machine can guess a variable assignment in polytime
- it can check whether the formula evaluates to true in polytime
- but if it accepts a satisfying assignment, it decides SAT
- we would need to change the acceptance condition: accept if *all* guesses are accepting.

Co-NP

Definition 10.2

A problem is in Co-NP, if its complement is in NP.

Theorem 10.3

1. $P \subseteq Co - NP$
2. *If* $P = NP$, *then* $P = NP = Co - NP$.

Proof.

Because P is closed under complementation. $\square$

More on TAUT

Theorem 10.4

If TAUT is in P, then every NP-Problem is in P.

Proof.

- a formula ϕ is satisfiable if $\neg\phi$ is not a tautology.
- Solve SAT in polytime:
  - convert ϕ to $\neg\phi$
  - run TAUT on $\neg\phi$
  - flip the result

$\square$

Question

- We have *not* given a polytime reduction from SAT to TAUT.
- Have we really shown that TAUT is NP-hard?

Cook Completeness

Definition 10.5

A problem X is Cook-NP-hard (complete), if one can show that if $X \in P$, then $P = NP$ (and $X \in NP$).

Example 10.6

We have shown that TAUT is Cook-NP-hard.

Definition 10.7

A problem X is Karp-NP-hard (complete), if every NP-Problem can be reduced to X in polytime (and $X \in NP$).

Remark

- Cook-completeness is Cook's original definition
- Cook was interested in why TAUT is hard
- TAUT as 'true mathematical theorems'
- We have used Karp completeness . . . why?

Cook vs Karp

Biggest Difference

- Cook lets us flip the answer after a polytime reduction
- Karp-completeness implies Cook completeness
- If P $=$ NP, they would both be the same

Why Karp?

- I have a deterministic algorithm for an NP-complete problem that runs in $\mathcal{O}(n^{\log n})$ time
- (or in time that is worse than poly, but not yet exponential)
- with Karp, I can solve any NP-Problem in that time
- with Cook, I cannot conclude anything.

Optimisation Problems

In (our) Theory:

- We have just considered yes/no problems

In Practice:

- We are looking for a solution
- e.g. a satisfying assignment
- or the size of the smallest node cover

Observation

- If we can solve the optimisation problem, we can solve the yes/no problem.

Example 10.8

- Yes/No problem: Does G have a node cover of size $\leq k$?
- Optimisation problem: What is the size of the smallest node cover for G?

Completeness for Optimisation Problems

Optimisation Problems

- cannot be in NP, as they are not yes/no problems
- also, want deterministic answer!
- but it makes sense to Cook-reduce the yes/no version to the optimisation version
- i.e if optimisation problem is polytime, then so is the yes/no version

Conclusion

If $P \neq NP$, then we cannot solve the optimisation version of a problem in polytime, if the yes/no version is NP-complete.