Algorithms and Complexity

Spring 2018
Aaram Yun

This page is intentionally left blank

Today

- NP-completeness

Proving hardness

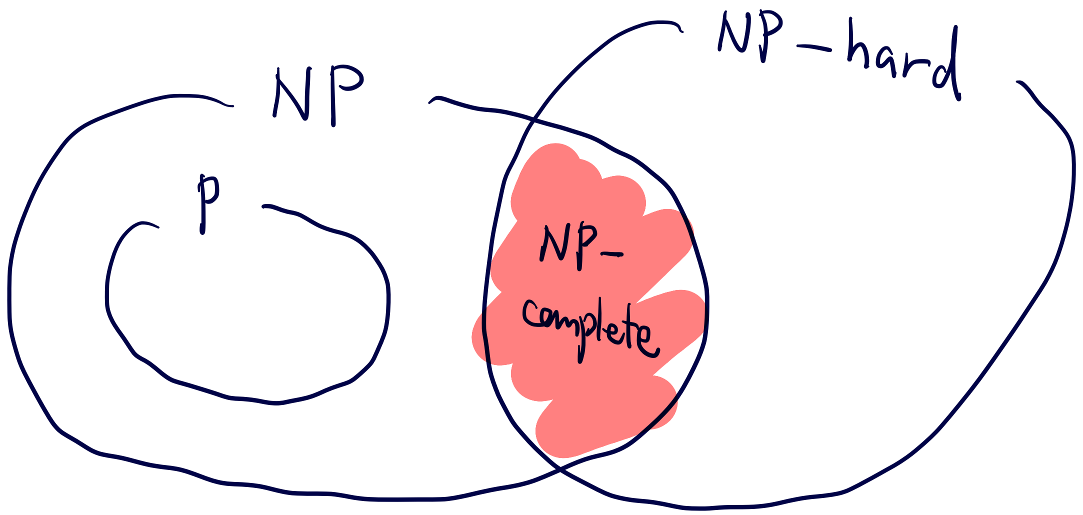
- The problem of proving that a problem is hard seems hard
- How about proving that a problem H is hard, *assuming that* $\mathcal{P} \neq \mathcal{NP}$?
- Contrapositively: if H can be solved efficiently, then *any* problem in $\mathcal{NP}$ can be solved efficiently
- Such a thing can be formulated and shown by reductions

Which reduction?

- Cook reductions are perfectly fine for the job
- Although, historical reasons and simplicity gave definitions based on Karp reductions (and Levin reductions, for search problems)

NP-complete problems

- A decision problem Π is $\mathcal{NP}$-complete iff
  - $\Pi \in \mathcal{NP}$, and (π is in NP)
  - Any problem $\Pi' \in \mathcal{NP}$ is Karp-reducible to Π (π is NP-hard)
- If Π is $\mathcal{NP}$-complete, then it is the hardest problem in $\mathcal{NP}$, in a sense



Pictorially

(Although, if P=NP, we have P=NP=NP-complete)

NP-complete search problems

- A search problem Π is $\mathcal{PC}$-complete iff
  - $\Pi \in \mathcal{PC}$, and
  - Any problem $\Pi' \in \mathcal{PC}$ is Levin-reducible to Π
- If R is $\mathcal{PC}$-complete, then S_R is $\mathcal{NP}$-complete

NP-hardness

- A decision problem Π is $\mathcal{NP}$-hard, if any problem $\Pi' \in \mathcal{NP}$ is Karp-reducible to Π
- A search problem Π is $\mathcal{PC}$-hard, if any problem $\Pi' \in \mathcal{PC}$ is Levin-reducible to Π

Abusing terminologies

- Sometimes, a search problem is referred as NP-complete, when it is in fact $\mathcal{PC}$-complete
  - And as NP-hard, when it is in fact $\mathcal{PC}$-hard

Existence

- Defining a unicorn doesn't produce a unicorn immediately!
- Having a definition of NP-completeness doesn't necessarily mean that an actual NP-complete problem exists
- In fact, there exist~~s~~ NP-complete problems
    - In fact, many examples of natural and interesting NP-complete problems exist
- Let's start with somewhat unnatural ones

Existence of NP-complete problems

- Theorem) There exist NP-complete relations and sets
  - Problem instances: $\bar{x} = \left\langle M, x, 1^t \right\rangle$
  - $R_{\mathtt{u}} := \{(\left\langle M, x, 1^t \right\rangle, y) \;:\; M \text{ accepts } (x, y) \text{ in } t \text{ steps } \& \; |y| \leq t\}$
  - $S_{\mathtt{u}} := \{\bar{x} \;:\; \exists y \text{ s.t. } (\bar{x}, y) \in R_{\mathtt{u}}\}$

Existence of NP-complete problems

- $R_{\mathbf{u}}$ is in $\mathcal{PC}$ (Hence, $S_{\mathbf{u}} \in \mathcal{NP}$)

Given $\bar{x} = \langle M, x, 1^t \rangle$ and y

do we have 1) $|y| \leq t$, and

2) M accepts (x, y) within t steps

Existence of NP-complete problems

- $R_{\mathtt{u}}$ is $\mathcal{PC}$-hard (Hence, $S_{\mathtt{u}}$ is $\mathcal{NP}$-hard)

Proving NP-completeness

- If you want to prove that a problem Π is NP-complete, then
  - First, prove that Π is in $\mathcal{NP}$ (or, in $\mathcal{PC}$ if it is a search problem), and
  - Second, pick your favorite NP-complete problem Π', and reduce Π' to Π