Algorithms and Complexity

Spring 2018

Aaram Yun

This page is intentionally left blank

Today

» Equivalence between search-based formulation and decision-based formulation

P versus NP? PF versus PC?

- Two formulations are equivalent!
- Theorem: $\mathcal{NP} = \mathcal{P} \iff \mathcal{PC} \subseteq \mathcal{PF}$
- So, conversely, $\mathcal{NP} \neq \mathcal{P} \iff \mathcal{PC} \nsubseteq \mathcal{PF}$

Backward direction

- If $\mathcal{PC} \subseteq \mathcal{PF}$ then $\mathcal{NP} \subseteq \mathcal{P}$
- Proof: this part is straightforward

proof) If $S \in NP$, $\exists R \in PC$ s.t. $S = S_R$

$\rightarrow R \in PC \subseteq PF$

$\rightarrow A$: eff. TM which solves R

$\forall x$
if $R(x) = \phi$
$\rightarrow A(x) = \perp$
if $R(x) \neq \phi$
$\rightarrow A(x) = y \in R(x)$

$\rightarrow \exists B$: eff. TM which solves S

$S \in P$

(B runs A. If A outputs $\perp$, output 0
If A outputs $y \in \{0,1\}^*$, output 1)

Forward direction

$x \in S_R$

- If $\mathcal{NP} \subseteq \mathcal{P}$ then $\mathcal{PC} \subseteq \mathcal{PF}$
- Proof: this part is slightly trickier
  - Assuming $\mathcal{NP} \subseteq \mathcal{P}$, and if $R \in \mathcal{PC}$ and $x \in \{0,1\}^*$, we need to show how to find some $y \in R(x)$ efficiently
  - Easy to check if such y exists or not (due to $\mathcal{NP} \subseteq \mathcal{P}$)
  - But how to actually *find* one such y? (or, how to find something when you can make only yes/no questions)

Forward direction

- Answer: binary search!
  - Consider "Twenty questions"
  - You ask only yes/no questions, but can find something, after all

NP-question.

"Is there any y s.t. $(x,y) \in R$?"

Yes

No

"Is there any y s.t. $(x,y) \in R$ **and** y starts with a 0?"

output $\perp$.

Yes

No

$\vdots$

$\vdots$

$R \in PC$

$|y| \leq p(|x|)$

R.

$S_R = \{x \mid \exists y,\ (x, y) \in R\}$ is not enough. (even though $S_R \in NP \subseteq P$)

$S'_R = \{(x, z) \mid \exists y \in \{0,1\}^*,\ (x, zy) \in R\}$.

We can see that $S'_R \in NP \subseteq P$.

$\rightarrow \exists A$: efficient TM solving S'_R.

If $(x, z) \notin S'_R$, then $A(x, z) = 0$

If $(x, z) \in S'_R$, then $A(x, z) = 1$.

We can construct B: eff. TM solving R.

$B(x) :=$ Run $A(x, \varepsilon)$ $\quad$ ($(x,\varepsilon) \in S_R' \iff \exists y, (x,y) \in R$)

If $A(x,\varepsilon) = 0$, then return $\perp$.

Let $z \leftarrow \varepsilon$

do

If $A(x, z0) = 1$ then

$z \leftarrow z0$

else

$z \leftarrow z1$

until $(x,z) \in R$

return z.

Why decision problems?

- Traditionally, complexity theory considered mostly decision problems
  - $\mathcal{P}$ versus $\mathcal{NP}$ problem, not $\mathcal{PF}$ versus $\mathcal{PC}$ problem
  - Even though search problems are more 'useful' than decision problems
- The equivalence is one reason why