Teorema de Ladner (assumindo ETH)

Nicollas Sdroievski

8 de Abril de 2019

1 Ideia Geral

Nesse encontro vamos provar uma versão relaxada do Teorema de Ladner, enunciado a seguir.

Teorema 1. *Se* $\mathsf{P} \neq \mathsf{NP}$*, então existem problemas em* $\mathsf{NP} \setminus \mathsf{P}$ *que não são* NP*-completos.*

Chamamos os problemas em $\mathsf{NP} \setminus \mathsf{P}$ que não são NP-completos de NP-intermediários. A prova completa desse teorema pode ser encontrada no capítulo 3 do Barak-Arora. Para enunciar a versão que vamos provar, apresentamos a Hipótese de Tempo Exponencial (*Exponential Time Hypothesis* ou somente ETH), que indica que é necessário tempo exponencial para decidir o problema SAT.

Hipótese 1. *(ETH) Existe um valor de* $\delta > 0$ *tal que* $\mathsf{SAT} \notin \mathsf{DTIME}(2^{\delta n})$.

O teorema que vamos provar então é o seguinte.

Teorema 2. *Assumindo a ETH, existem problemas em* $\mathsf{NP} \setminus \mathsf{P}$ *que não são* NP*-completos.*

A prova é baseada na aula 14 do curso de complexidade computacional da Carnegie Mellon University (link aqui).

2 Demonstração

Vamos mostrar que a seguinte linguagem L é NP-intermediária.

$$L = \{\langle \phi, 1^{2^{\sqrt{|\phi|}}} \rangle \mid \phi \in \mathsf{SAT}\}.$$

Para isso precisamos provar primeiro que $L \in \mathsf{NP}$ (**Exercício**).

Agora provamos que $L \notin \mathsf{P}$. Assuma por contradição que $L \in \mathsf{P}$, vamos mostrar um algoritmo subexponencial para SAT, contradizendo a ETH. Seja M a MT que decide L em tempo N^c para algum $c > 0$. Considere o seguinte algoritmo para SAT:

1. Na entrada ϕ com $|\phi| = n$.
2. Produza a string $y = \langle \phi, 1^{2^{\sqrt{n}}} \rangle$.
3. Retorne $M(y)$.

Exercício: prove que esse algoritmo decide SAT e que seu tempo de execução é $O(2^{\sqrt{n}})$. Logo, $\mathsf{SAT} \in \mathsf{DTIME}(2^{c\sqrt{n}})$, como $2^{c\sqrt{n}} = o(2^{\delta n})$ para qualquer δ, isso contradiz a ETH. Concluímos que $L \notin \mathsf{P}$.

Agora vamos mostrar, de maneira semelhante (mostrando um algoritmo rápido demais para SAT), que L não é NP-completa. Antes disso, mostramos um algoritmo que decide L em tempo subexponencial.

1. Na entrada x com $|x| = N$.
2. Verifique se $x = \langle \phi, 1^{2^{\sqrt{|\phi|}}} \rangle$.
3. Obtenha ϕ de tamanho n.
4. Execute um algoritmo de força bruta para decidir se $\phi \in \mathsf{SAT}$.

É possível executar o passo 2 em tempo polinomial no valor de N. Perceba que no passo 3, temos que $n \leq \log^2 N$. Assim, no passo 4, o algoritmo de força bruta executa em tempo

$$O(2^n) = O(2^{\log^2 N}) = O(N^{\log N}).$$

Agora, assuma por contradição que L é NP-completa, e seja f a redução de SAT para L. Note que o tempo de execução de f é no máximo n^k. Considere o seguinte algoritmo para SAT.

1. Na entrada ϕ com $|\phi| = n$.
2. Compute $y = f(\phi)$.
3. Execute o algoritmo para decidir L com a entrada y.

Exercício: prove que o esse algoritmo decide SAT. Perceba que no passo 2 o tamanho de y é no máximo n^k. Assim, o tempo total de execução do algoritmo fica dominado pelo passo 3, que leva tempo

$$O((n^k)^{\log(n^k)}) = O(n^{k^2 \log n}).$$

Como novamente $n^{k^2 \log n} = o(2^{\delta n})$ para qualquer δ, contradizemos a ETH. Concluímos que L não é NP-completa, e por fim que L é NP-intermediária.