CS 121 Lecture 13

Oct 15, 2019

Efficient Computation

Key takeaways

- some problems have trivial/native/brute force solutions of exponential time algorithms ($2^{poly(n)}$ or $exp(n)$)
- some have much faster polynomial time algorithms (poly(n) often $O(n)$ or $O(n^2)$)
- poly vs exponential may be more imp than computable vs uncomputable

Measuring Running Time

* $2^{O(n)}$ vs $O(2^n)$

$2^{const \cdot n}$ versus $const \cdot 2^n$ ← tighter bound

- roughly # of steps taken in the computational model

within polynomial time of each other:
- NAND-TM (# lines of code)
- TM (# of transitions)
- RAM / Python / ... similar notions

Solving Linear Eqns

Input: n linear equations in n variables: $Ax=b$, with $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$.

Gaussian elimination is polynomial time.

$$\begin{aligned} 3y + 2z &= 4 \\ x + y + z &= 1 \\ x + 2y &= 1 \end{aligned} \rightarrow \begin{bmatrix} 0 & 3 & 2 \\ 1 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix}$$

$$T(n) = T(n-1) + O(n^2) \rightarrow T(n) = O(n^3)$$

Optimization

Given function $f: K \rightarrow \mathbb{R}$, find the minimum

Linear Programming

Compute $\min_{x \in K} f(x)$ where f is linear and K is polytope:

$K = \{x \in \mathbb{R}^n \mid Ax \leq b\}$

Theorem: There is a poly(n) time algorithm for LP on n variables.

Integer Programming

Compute $\min_{x \in I} f(x)$ where f is linear and $I = K \cap \mathbb{Z}^n$;

$K = \{x \mid Ax \leq b\}$ (I integer points in polytope K)

No known polynomial-time algorithm for integer programming.