Computational Complexity

Lecture 9: Probabilistic Algorithms

Ronald de Haan
me@ronalddehaan.eu

University of Amsterdam

May 6, 2024
Recap

- Non-uniform complexity
- Circuit complexity
- TMs that take advice
- The Karp-Lipton Theorem: if $NP \subseteq P/poly$, then $\Sigma^p_2 = \Pi^p_2$
What will we do today?

- Probabilistic algorithms
- Complexity classes BPP, RP, coRP, ZPP
Randomized algorithms

- Randomized (or probabilistic) algorithms are a realistic extension of deterministic algorithms.
- They have access to a random number generator (or random coin flips).
- The outcome of such algorithms is a random variable.
- The running time of such algorithms is a random variable.
Input: you’re given $m \in \mathbb{N}$ and you have access to an oracle $O$ that can give you a value $O(i) \in \{a, b\}$, for each $i \in \{1, \ldots, 2^m\}$

Promise: for exactly half of the $i$’s it holds that $O(i) = a$, and so for the other half, $O(i) = b$

Task: output some $i \in \{1, \ldots, 2^m\}$ such that $O(i) = a$

When we consider deterministic (non-randomized) algorithms, what worst-case running time (and # of oracle queries) can we achieve for this problem?

- We need $2^m/2 = 2^{m-1}$ queries in the worst case, and $\Theta(2^m)$ time
Monte Carlo algorithm

\[ i := 0; \]
\[ \text{while } i < k \text{ do} \]
\[ \quad \text{randomly pick } j \in \{1, \ldots, 2^m\}; \]
\[ \quad \text{query the oracle: } o_j := O(j); \]
\[ \quad \text{if } o_j = a \text{ then} \]
\[ \quad \quad \text{return } j; \]
\[ \quad \text{else} \]
\[ \quad \quad i := i + 1; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{randomly pick } j \in \{1, \ldots, 2^m\}; \]
\[ \text{return } j; \]

- Runs for \( k \) rounds, so takes time \( O(k \cdot m) \)
- Probability of a correct answer: \( 1 - (1/2)^{k+1} \)
- Works for any value of \( k \)
- The running time does not vary randomly
- Non-zero error probability
Las Vegas algorithm

\[
\text{while } True \text{ do}
\begin{align*}
\text{randomly pick } j \in \{1, \ldots, 2^m\}; \\
\text{query the oracle: } o_j := O(j); \\
\text{if } o_j = a \text{ then} \\
\quad \text{return } j;
\end{align*}
\text{end}
\]

- The running time varies randomly (and is polynomial in expectation)
- Zero error probability

- Probability of a correct answer (given that it halted): 1

- Expected running time $O(m)$:

\[
O(m) \cdot \left[ 1 \cdot \frac{1}{2} + 2 \cdot \left( \frac{1}{2} \right)^2 + 3 \cdot \left( \frac{1}{2} \right)^3 + \cdots \right] = O(m)
\]

because

\[
\lim_{n \to \infty} \sum_{i=1}^{n} \frac{i}{2^i} = 2
\]
**Definition**

*Probabilistic Turing machines (PTM)* are variants of (deterministic) TMs, where:

- There are two transition functions $\delta_1, \delta_2$.
- At each step, one of $\delta_1, \delta_2$ is chosen randomly, both with probability $1/2$. (Each such choice is made independently.)
- (As halting states, it has an accept state $q_{\text{acc}}$ and a reject state $q_{\text{rej}}$.)

- $\mathbb{M}(x)$ denotes the random variable corresponding to the output of $\mathbb{M}$ on input $x$.
- $\mathbb{M}$ runs in time $T(n)$ if for every input $x$ and every sequence of nondeterministic choices, $\mathbb{M}$ halts within $T(|x|)$ steps, regardless of the random choices made.
Definition (BPTIME)

Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A problem $L \subseteq \{0, 1\}^*$ is in $\text{BPTIME}(T(n))$ if there exists a PTM $M$ that runs in time $O(T(n))$, such that for each $x \in \{0, 1\}^*$:

$$\mathbb{P}[M(x) = L(x)] \geq \frac{2}{3},$$

where $L(x) = 1$ if $x \in L$, and $L(x) = 0$ if $x \notin L$.

- **BP**: Bounded-error Probabilistic
- These are *Monte Carlo algorithms with two-sided (bounded) error*

Definition (BPP)

$$\text{BPP} = \bigcup_{c \geq 1} \text{BPTIME}(n^c).$$
Theorem

A problem $L \subseteq \{0, 1\}^*$ if and only if there exists a polynomial-time deterministic TM $M$ and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for each $x \in \{0, 1\}^*$:

$$\mathbb{P}_{r \in_R \{0,1\}^{p(|x|)}} \left[ M(x, r) = L(x) \right] \geq 2/3.$$  

(Here $\in_R$ denotes (sampling from) the uniform distribution.)

- This is analogous to the verifier definition of NP
  - Using a probabilistic interpretation of the certificates, rather than existentially quantifying over them
One-sided error: RP and coRP

Definition (RTIME)
Let $T : \mathbb{N} \rightarrow \mathbb{N}$ be a function. A problem $L \subseteq \{0, 1\}^*$ is in $\text{RTIME}(T(n))$ if there exists a PTM $M$ that runs in time $O(T(n))$, such that for each $x \in \{0, 1\}^*$:

- if $x \in L$, then $\mathbb{P}[M(x) = 1] \geq 2/3$,
- if $x \notin L$, then $\mathbb{P}[M(x) = 0] = 1$.

These are *Monte Carlo algorithms with one-sided (bounded) error*.

Definition (RP)

$\text{RP} = \bigcup_{c \geq 1} \text{RTIME}(n^c)$.
One-sided error: RP and coRP (ct’d)

**Definition (coRTIME)**

Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A problem $L \subseteq \{0, 1\}^*$ is in coRTIME($T(n)$) if there exists a PTM $M$ that runs in time $O(T(n))$, such that for each $x \in \{0, 1\}^*$:

- if $x \in L$, then $P[M(x) = 1] = 1$,
- if $x \not\in L$, then $P[M(x) = 0] \geq 2/3$.

These are also *Monte Carlo algorithms with one-sided (bounded) error*

**Definition (coRP)**

$$\text{coRP} = \bigcup_{c \geq 1} \text{coRTIME}(n^c),$$

or equivalently: $\text{coRP} = \{ \overline{L} \mid L \in \text{RP} \}$. 
Definition (expected running time)

Let $T : \mathbb{N} \rightarrow \mathbb{N}$ be a function and let $M$ be a PTM. Then $M$ runs in *expected time* $T(n)$, if for each $x \in \{0, 1\}^*$ it holds that $\mathbb{E}[\text{time}_M(x)] \leq T(|x|)$.

Definition (ZPTIME)

Let $T : \mathbb{N} \rightarrow \mathbb{N}$ be a function. A problem $L \subseteq \{0, 1\}^*$ is in $\text{ZPTIME}(T(n))$ if there exists a PTM $M$ that runs in expected time $O(T(n))$, such that for each $x \in \{0, 1\}^*$, whenever $M$ halts on $x$ then $M(x) = L(x)$.

- These are *Las Vegas algorithms*

Definition (ZPP)

$$ZPP = \bigcup_{c \geq 1} \text{ZPTIME}(n^c).$$
We used the constant $\frac{2}{3}$ in the definitions of BPP, etc.

In fact, each constant $> \frac{1}{2}$ would work, and even $> \frac{1}{2} + |x|^{-c}$.

We can make the error probability very small

**Theorem (Error reduction for BPP)**

Let $L \subseteq \{0, 1\}^*$ be a decision problem, and suppose that there exists a polynomial-time PTM $M$ such that for each $x \in \{0, 1\}^*$, $\Pr[M(x) = L(x)] \geq \frac{1}{2} + \frac{1}{|x|^c}$.

Then for every constant $d > 0$, there exists a polynomial-time PTM $M'$ such that for each $x \in \{0, 1\}^*$, $\Pr[M'(x) = L(x)] \geq 1 - \frac{1}{2^{|x|^d}} = 1 - 2^{-|x|^d}$.

Idea: run $M$ many times and output the majority answer
Some relations

- \( \text{RP} \subseteq \text{BPP} \), \( \text{coRP} \subseteq \text{BPP} \)
- \( \text{RP} \subseteq \text{NP} \), \( \text{coRP} \subseteq \text{coNP} \)
  - Homework!
- \( \text{ZPP} = \text{RP} \cap \text{coRP} \)
  - Homework!
- \( \text{BPP} \subseteq \text{P/poly} \)
  - Idea: by using error reduction, you can find some \( r \in \{0, 1\}^{p(n)} \) for each \( n \) that can be used as “certificate” to give the correct answer for each \( x \in \{0, 1\}^n \).
- \( \text{BPP} \subseteq \Sigma^p_2 \), \( \text{BPP} \subseteq \Pi^p_2 \)
- Probabilistic algorithms
- Complexity classes BPP, RP, coRP, ZPP
Approximation algorithms

The PCP Theorem