Overview:

Last time:
1. More Paradigms for Randomized Algorithms
2. Complexity Classes for Randomized Algorithms (PP & BPP)

Today:
1. More Complexity Classes

1 More Complexity Classes

1.1 RP: Randomized Polynomial time

**Solovay-Strassen Primality Test:**
If \( n \) is an odd prime, \( \forall a \), it satisfies the following equation:
\[
 a^{n-1} \equiv (\frac{a}{n}) \mod n
\]
Furthermore, if \( n \) is not prime, this equation fails with probability \( \geq \frac{1}{2} \).

In other words, Solovay-Strassen primality test always accepts prime numbers, and rejects composite numbers at least half of the time.

**Definition**

L \( \in \) RP (Randomized Polynomial time) if \( \exists \) polynomial time randomized algorithm \( A \), s.t.
\[
 x \in L \Rightarrow Pr(A(x) \text{ accepts}) \geq \frac{1}{2} \\
 x \notin L \Rightarrow Pr(A(x) \text{ accepts}) = 0
\]
In the case of Solovay-Strassen primality test,
\[
 \text{composites} \in RP \\
 \text{primes} \in coRP
\]
where coRP represents the complement complexity class of randomized polynomial.
1.2 ZPP: Zero-error Probabilistic Polynomial time

Quick Sort Algorithm (runtime randomized):
1. Pick random \(x \in L\);
2. Compare it with everything else;
3. Sort elements \(< x\);
4. Sort elements \(> x\)

For the runtime of Quick Sort, we have:
\[
E(\text{runtime}) = O(n \log n)
\]
\[
\text{runtime in worst case} = O(n^2)
\]

Definition \(L \in \text{ZPP}\) (Zero-error Probabilistic Polynomial time) if \(\exists\) randomized algorithm \(A\), s.t.
\[
Pr(A(x) \text{ accepts } \iff x \in L) = 1
\]

And \(\forall x\),
\[
E[\text{runtime of } A(x)] = \text{poly}(|x|)
\]

Theorem \(ZPP = \text{RP} \cap \text{coRP}\)

Proof
1. \(ZPP \supseteq \text{RP} \cap \text{coRP}\)

We can use a RP algorithm and a coRP algorithm to construct the following algorithm satisfying the requirements for ZPP:

Repeat:
1. Run RP algorithm: If accept \(\Rightarrow\) Return YES
2. Run coRP algorithm: If reject \(\Rightarrow\) Return NO

According to the definition of RP and coRP, our algorithm has \(\geq \frac{1}{2}\) chance to return a zero-error result after each iteration. It satisfies the requirements for ZPP.

2. \(ZPP \subseteq \text{RP} \cap \text{coRP}\)

Here, we simply show that \(ZPP \subseteq \text{RP}\). Since ZPP is closed under complement (which can be seen from switching acceptance and rejection in the definition), we can also get \(ZPP \subseteq \text{coRP}\).

Suppose \(A\) is a zero-error algorithm that runs in polynomial expected-time \(T\) (i.e. \(A\) is a ZPP algorithm and \(E[\text{runtime}(A)] = \text{polynomial time} = T\)). We will now use this algorithm \(A\) to construct algorithms satisfying the requirements for RP.

Consider the following RP algorithm \(A'\): it runs the algorithm \(A\) for \(10T\) time. If \(A\) terminates, then we can return the output of \(A\). If \(A\) does not terminate (with probability \(\leq \frac{1}{10}\)), then we simply reject.

Note that this new algorithm \(A\) always rejects inputs not in \(L\). Moreover, if input \(x \in L\), then \(A\) would reject \(x\) only if the algorithm \(A\) ran for more than \(10T\) time. However, since the expected run time for \(A\) was only \(T\), the probability that \(A\) ran for more than \(10T\) time is less than a half. Thus \(A\) rejects strings in \(L\) with probability less than \(\frac{1}{2}\), which satisfies the requirements for RP.
1.3 More about BPP (Adleman’s Theorem)

Recall, algorithm in BPP (Bounded-error Probabilistic Polynomial time) can improve error probability to \( \epsilon \) with running on \( O(lg(\frac{1}{\epsilon})) \) many different random strings. \( 2^n \) strings of length \( n \) with polynomial-time blow up reduce error probability to \( 4^{-n} \).

\[ \exists \text{ some set of } O(n) \text{ random strings } r_1, r_2, ..., r_m \text{ such that the majority answer of } A(x, r_1), A(x, r_2), ..., A(x, r_m) \text{ is always right for all inputs } x \text{ of length } n. \]

If we hardcoded \( r_1, r_2, ..., r_m \) in our algorithm, we have a deterministic polynomial-time algorithm.

**Adleman’s Theorem (2.9 in book):** BPP \( \subset \) P/poly

**Question:** Does Randomization help?
- No (Adleman)
- Yes (restricted models of computation)
- Maybe (does P=BPP?)

Submitted by Liu, Yuanchen
Li, Zhefeng on October 4, 2017.