Advanced complexity analysis

Textbook: Chapter 10

Probabilistic Algorithms

Def: A **probabilistic Turing machine** M is an NTM where each "coin-flip" step has 2 legal equally-probable next moves. Every branch b of M has a probability defined by:

$$\Pr[b] = 2^{-k}$$

where k is the number of coin-flip steps in b. The probability that M accepts w is

$$\Pr[M \texttt{ accepts } w] = \sum_{b \texttt{ accepts } w} \Pr[b]$$

We say M **accepts** A **with error probability** ϵ if:

1. $w \in A$ implies $\Pr[M \texttt{ acccepts } w] \geq 1 - \epsilon$ and
2. $w \notin A$ implies $\Pr[M \texttt{ rejects } w] \geq 1 - \epsilon$

This means the probability of M being wrong is at most ϵ.

Probabilistic Complexity Classes

Def: *BPP* (Bounded-Error Probabilistic Polynomial) is the class of languages which are recognized by probabilistic polynomial-time Turing machines with an error probability of $\frac{1}{3}$ (or equivalently any other constant c where $0 < c < \frac{1}{2}$).

BPP	Actually F	Actually T
Answered F	$\geq 1 - c$	$< c$
Answered T	$< c$	$\geq 1 - c$

Probabilistic Complexity Classes Pt. 2

Def: RP (Randomized Polynomial) is the class of languages recognized by probabilistic polynomial-time Turing machines where any strings in the language are accepted with a probability $\geq \frac{1}{2}$ and any strings not in the language are rejected with a probability of 1.

RP	Actually F	Actually T
Answered F	1	$< \frac{1}{2}$
Answered T	0	$\geq \frac{1}{2}$

Example: Primality and Fermat's Little Theorem

- A number c is prime if $\forall a, b[ab \neq c]$
- **Note**: Primality is now known to be $\in P$
- We say a and b are **equivalent modulo** p if $\exists k[a = b + kp]$ (they differ by a multiple of p)
    - Then we say $a \equiv b \pmod{p}$
- Let $x \mod p = y$ mean that y is the smallest integer such that $x \equiv y \pmod{p}$
- Let $\mathcal{Z}_p^+$ be the set of nonnegative integers below p

Thm: Fermat's little theorem. If p is prime and $a \in \mathcal{Z}_p^+$, then $a^{p-1} \equiv 1 \pmod{p}$.

Primality Pt. 2

Corollary: By contraposition: if $a^{p-1} \not\equiv 1 \pmod p$ and $a \in \mathcal{Z}_p^+$, then p is not prime. This is called a **Fermat test**.

- This does not imply the converse!
- A number p is **pseudoprime** if it is composite and $a^{p-1} \not\equiv 1 \pmod p$ for some a
  - a is called a **liar**
  - A randomly chosen a is a liar at most $\frac{1}{2}$ of the time

Primality Pt. 3

```python
import random

# Return whether p is probably prime
# Probability of p being composite
# and pseudoprime is 2 ** -k
def pseudoprime(p, k) -> bool:
    for i in range(k):
        # Random uniform integer 1 <= a < p
        a = random.randrange(1, p)
        if (a ** (p - 1)) % p != 1:
            return False
    return True
```

Primality Pt. 4

```python
def prime(p, k) -> bool:
    if p % 2 == 0:
        return p == 2  # Any non-2 evens
    for i in range(k): # k Fermat tests
        a = random.randrange(1, p)
        if (a ** (p - 1)) % p != 1:
            return False  # Fermat test
        h = 0  # h is the exponent of 2
        while (p // (2 ** h)) % 2 == 0:
            h += 1
        for e in range(h, 0, -1):
            cur = a ** (2 ** e) % p
            if cur != 1:
                if cur != p - 1:
                    return False
                break
    return True
```

Alternation and Alternating Turing Machines

- A nondeterministic Turing Machine accepts if *any* of its branches do
  - This is not the only way!
- We could also specify that *all* branches must
- We could even alternate between *all* and *any*

Def: An **alternating Turing Machine (ATM)** is a nondeterministic Turing machine where every non-terminal state (not accepting or rejecting) is either **universal** or **existential**. A *universal* node in the nondeterministic execution tree accepts if **all** of its nondeterministic branches do. An *existential* node, on the other hand, accepts if **any** sub-branches do.

- Time and space complexity are defined as in nondeterministic Turing machines
- This allows us to do "short-circuit" boolean logic in special nondeterministic cases

Next up: Nothing!

Bonus topics you can find in the book in this chapter:

- P-Completeness
- Cryptography
- Public-Key Cryptosystems
- One-Way Functions
- Trapdoor Functions