Randomized Complexity Classes

- Let N be a polynomial-time precise NTM that runs in time $p(n)$ and has 2 nondeterministic choices at each step.
- N is a **polynomial Monte Carlo Turing machine** for a language L if the following conditions hold:
  - If $x \in L$, then at least half of the $2^{p(|x|)}$ computation paths of N on x halt with "yes."
  - If $x \notin L$, then all computation paths halt with "no."
- The class of all languages with polynomial Monte Carlo TMs is denoted **RP** for **randomized polynomial time**.

Comments on RP

- Nondeterministic steps can be seen as fair coin flips.
- There are no false positive answers.
- The probability of false negatives is at most 0.5.
- Any constant $0 \leq \epsilon \leq 1$ can replace 0.5.
  - By repeating the algorithm k times, the probability of false negatives can be reduced to $(1 - \epsilon)^k$.
  - Now pick $k = \lceil -\frac{1}{\log_2 1-\epsilon} \rceil$.
- In fact, ϵ can be arbitrarily close to 0 as long as it is of the order $1/p(n)$ for some polynomial $p(n)$.
  - $-\frac{1}{\log_2 1-\epsilon} = O(\frac{1}{\epsilon}) = O(p(n))$.

Where RP Fits

- $\text{P} \subseteq \text{RP} \subseteq \text{NP}$.
  - A polynomial-time deterministic TM is like a polynomial Monte Carlo TM except that all the coin flips are ignored.
  - A polynomial Monte Carlo TM is a polynomial-time NTM with extra demands on the number of accepting paths.
- COMPOSITENESS $\in$ RP.
- PRIMES $\in$ coRP.
- $\text{RP} \cup \text{coRP}$ is a "plausible" notion of efficient computation.

ZPP[a] (Zero Probabilistic Polynomial)

- The class **ZPP** is defined as RP $\cap$ coRP.
- A language in ZPP has *two* Monte Carlo algorithms, one with no false positives and another with no false negatives.
- If we repeatedly run both Monte Carlo algorithms, *eventually* one definite answer will come (unlike RP).
  - A *positive* answer from the one without false positives.
  - A *negative* answer from the one without false negatives.
- The algorithm is called **Las Vegas**.

[a] Gill, 1977.

The ZPP Algorithm

```
{Suppose that L ∈ ZPP.}
{N_1 has no false positives, and N_2 has no false
 negatives.}
while true do
    if N_1(x) = "yes" then
        return "yes";
    end if
    if N_2(x) = "no" then
        return "no";
    end if
end while
```

ZPP (continued)

- The *expected* running time for it to happen is polynomial.
  - The probability that a run of the 2 algorithms does not generate a definite answer is 0.5.
  - Let $p(n)$ be the running time of each run.
  - The expected running time for a definite answer is thus

$$\sum_{i=1}^{\infty} 0.5^i i p(n) = 2p(n).$$

- PRIMES $\in$ ZPP (whose proof remains inaccessible).

Me Too, RP?

```
{Suppose that L ∈ RP.}
{N decides L without false positives.}
while true do
    if N(x) = "yes" then
        return "yes";
    end if
    {What to do here?}
end while
```

- You eventually get a "yes" if $x \in L$.
- But how to get a "no" when $x \notin L$?

PP

- A language L is in the class **PP** if there is a polynomial-time precise NTM N such that:
  - For all inputs x, $x \in L$ if and only if more than half of the computations of N (i.e., $2^{p(n)-1} + 1$ or up) on input x end up with a "yes."
  - We say that N decides L by majority.
- MAJSAT: is it true that the majority of the 2^n truth assignments to ϕ's n variables satisfy it?
- MAJSAT is PP-complete.
- PP is closed under complement.

NP vs. PP

Theorem 61 *$NP \subseteq PP$.*

- Suppose that $L \in$ NP is decided by an NTM N.
- Construct a new NTM N':
  - N' has one more extra state s than N.
  - N' starts at s and either branches to N's program or simply accepts (after $p(|x|)$ steps).
- Consider an input x.
- Suppose that N on x computes for $p(|x|)$ steps and produces $2^{p(|x|)}$ computation paths.

The Proof (continued)

- Then N' has $2^{p(|x|)+1}$ computation paths.
- Half of these will always halt with "yes."
- Thus a majority of the paths of N' accept x if and only if at least one path of N accepts x.
- That is, if and only if $x \in L$.
- So N' accepts L by majority and $L \in \text{PP}$.

Theory of Large Deviations

- You have a *biased* coin.
- One side has probability $0.5 + \epsilon$ to appear and the other $0.5 - \epsilon$, for some $0 < \epsilon < 1$.
- But you do not know which is which.
- How to decide which side is the more likely—with high confidence?
- Answer: Flip the coin many times and pick the side that appeared the most times.
- Question: Can you quantify the confidence?

The Chernoff Bound

Theorem 62 (Chernoff, 1952) *Suppose that* $x_1, x_2, \ldots, x_n$ *are independent random variables taking the values 1 and 0 with probabilities* p *and* $1 - p$*, respectively. Let* $X = \sum_{i=1}^{n} x_i$*. Then for all* $0 \leq \theta \leq 1$*,*

$$\text{prob}[\, X \geq (1 + \theta)pn \,] \leq e^{-\theta^2 pn/3}.$$

- The probability that the deviate of a **binomial random variable** from its expected value decreases exponentially with the deviation.
- The Chernoff bound is asymptotically optimal.

The Proof

- Let t be any positive real number.
- Then

$$\text{prob}[\, X \geq (1+\theta)pn \,] = \text{prob}[\, e^{tX} \geq e^{t(1+\theta)pn} \,].$$

- Markov's inequality (p. 282) generalized to real-valued random variables says that

$$\text{prob}\left[\, e^{tX} \geq kE[\, e^{tX} \,] \,\right] \leq 1/k.$$

- With $k = e^{t(1+\theta)pn}/E[\, e^{tX} \,]$, we have

$$\text{prob}[\, X \geq (1+\theta)pn \,] \leq e^{-t(1+\theta)pn} E[\, e^{tX} \,].$$

The Proof (continued)

- Because $X = \sum_{i=1}^{n} x_i$ and x_i's are independent,

$$E[\, e^{tX}\,] = (E[\, e^{t x_1}\,])^n = [\, 1 + p(e^t - 1)\,]^n.$$

- Substituting, we obtain

$$\begin{aligned} \text{prob}[\, X \geq (1+\theta)pn\,] &\leq e^{-t(1+\theta)pn} [\, 1 + p(e^t - 1)\,]^n \\ &\leq e^{-t(1+\theta)pn} e^{pn(e^t - 1)} \end{aligned}$$

as $(1+a)^n \leq e^{an}$ for all $a > 0$.

The Proof (continued)

- With the choice of $t = \ln(1+\theta)$, the above becomes

$$\text{prob}[\, X \geq (1+\theta)pn \,] \leq e^{pn[\,\theta - (1+\theta)\ln(1+\theta)\,]}.$$

- The exponent expands to $-\frac{\theta^2}{2} + \frac{\theta^3}{6} - \frac{\theta^4}{12} + \cdots$ for $0 \leq \theta \leq 1$, which is less than

$$-\frac{\theta^2}{2} + \frac{\theta^3}{6} \leq \theta^2 \left(-\frac{1}{2} + \frac{\theta}{6} \right) \leq \theta^2 \left(-\frac{1}{2} + \frac{1}{6} \right) = -\frac{\theta^2}{3}.$$

Effectiveness of the Majority Rule

From $\text{prob}[\, X \leq (1-\theta)pn \,] \leq e^{-\frac{\theta^2}{2}pn}$ (prove it), it follows that:

Corollary 63 *If* $p = (1/2) + \epsilon$ *for some* $0 \leq \epsilon \leq 1/2$, *then*

$$\text{prob}\left[\sum_{i=1}^{n} x_i \leq n/2\right] \leq e^{-\epsilon^2 n/2}.$$

- The textbook's corollary to Lemma 11.9 seems incorrect.
- Our original problem (p. 329) hence demands $\approx 1.4k/\epsilon^2$ independent coin flips to guarantee making an error with probability at most 2^{-k} with the majority rule.

BPP[a] (Bounded Probabilistic Polynomial)

- The class **BPP** contains all languages for which there is a precise polynomial-time NTM N such that:
  - If $x \in L$, then at least $3/4$ of the computation paths of N on x accept, and
  - If $x \notin L$, then at least $3/4$ of the computation paths of N on x reject.
- N accepts or rejects by a *clear* majority.

---

[a] Gill, 1977.

Magic 3/4?

- The number 3/4 bounds the probability of a right answer away from 1/2.
- Any constant *strictly* between 1/2 and 1 can be used without affecting the class BPP.
- In fact, any 0.5 plus inverse polynomial

$$0.5 + 1/p(n)$$

between 1/2 and 1 can be used.

The Majority Vote Algorithm

Suppose that L is decided by N by majority $(1/2) + \epsilon$.

```
for i = 1, 2, ..., 2k + 1 do
    Run N on input x;
end for
if "yes" is the majority answer then
    "yes";
else
    "no";
end if
```

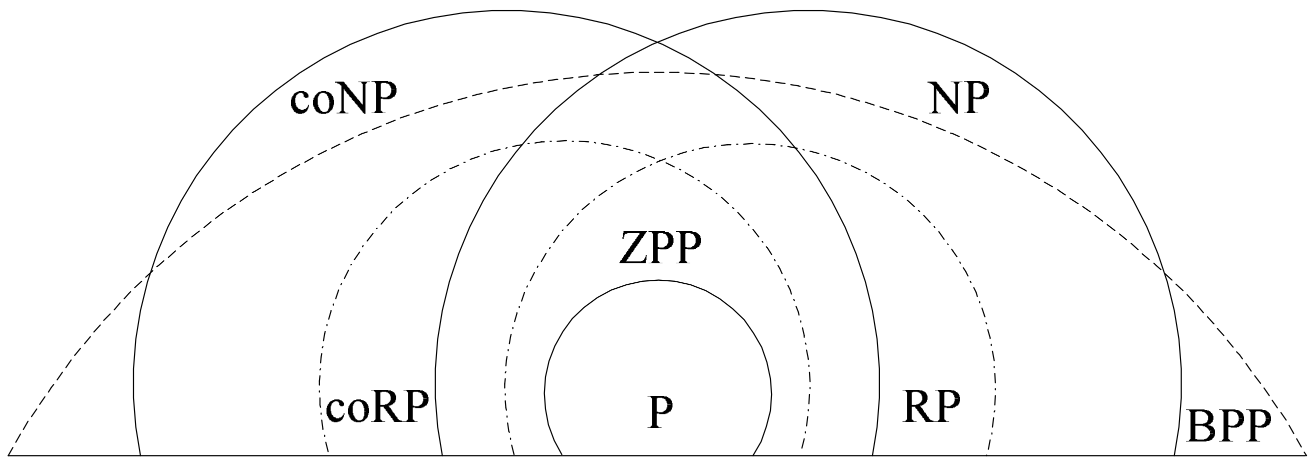
Analysis

- The running time remains polynomial, being $2k + 1$ times N's running time.
- By Corollary 63 (p. 334), the probability of a false answer is at most $e^{-\epsilon^2 k}$.
- By taking $k = \lceil 2/\epsilon^2 \rceil$, the error probability is at most $1/4$.
- As with the RP case, ϵ can be any inverse polynomial, because k remains polynomial in n.

Aspects of BPP

- BPP is the most comprehensive yet plausible notion of efficient computation.
  - If a problem is in BPP, we take it to mean that the problem can be solved efficiently.
- $(\mathrm{RP} \cup \mathrm{coRP}) \subseteq (\mathrm{NP} \cup \mathrm{coNP})$ and $(\mathrm{RP} \cup \mathrm{coRP}) \subseteq \mathrm{BPP}$.
- Whether $\mathrm{BPP} \subseteq (\mathrm{NP} \cup \mathrm{coNP})$ is unknown.

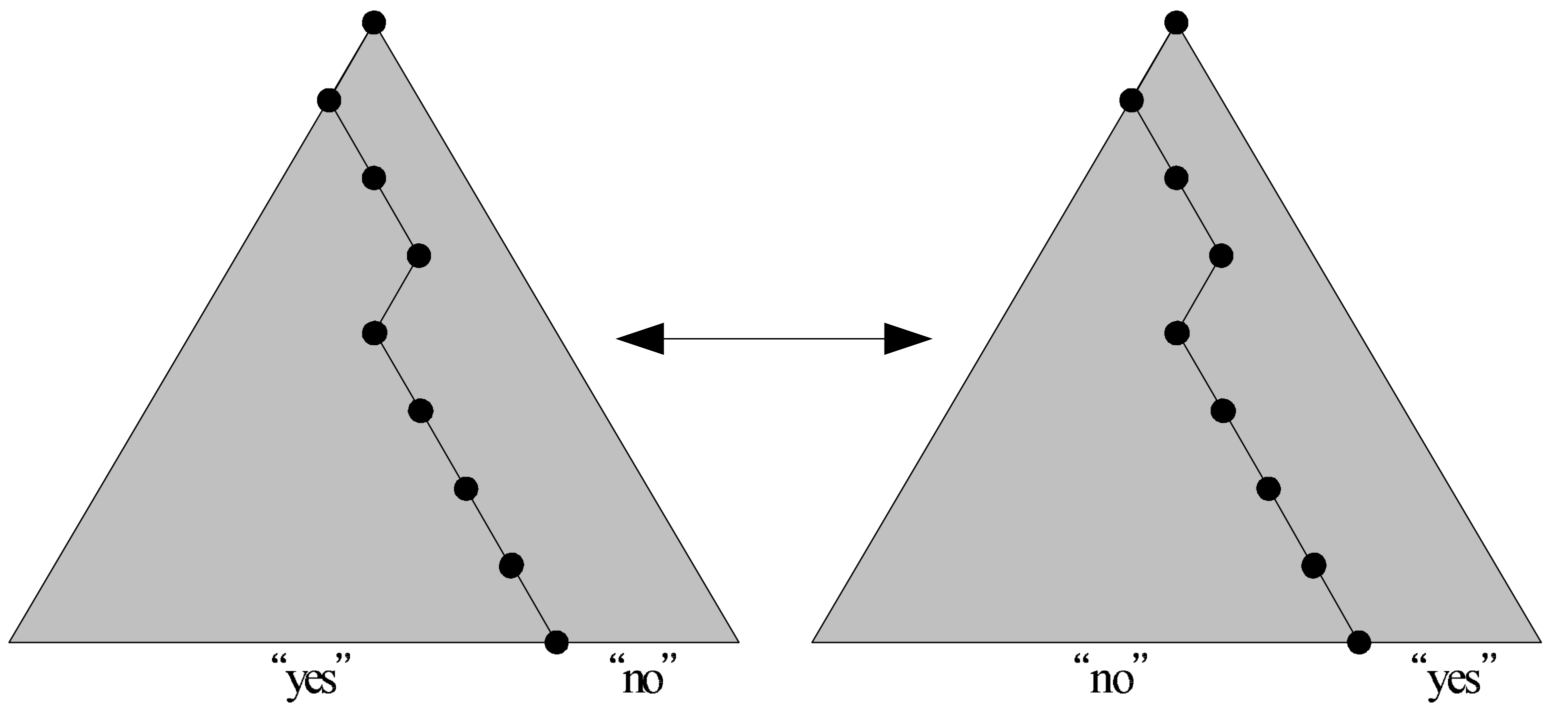
A Review of Classes



coBPP

- The definition of BPP is symmetric: acceptance by clear majority and rejection by clear majority.
- An algorithm for $L \in$ BPP becomes one for $\bar{L} \in$ coBPP by reversing the answer.
- Hence BPP $=$ coBPP.
- This approach does not work for RP (it did not work for NP either).

BPP and coBPP



Circuit Complexity

- Circuit complexity is based on boolean circuits instead of Turing machines.
- A boolean circuit with n inputs computes a boolean function of n variables.
- By identify `true` with 1 and `false` with 0, a boolean circuit with n inputs accepts certain strings in $\{0,1\}^n$.
- To relate circuits with arbitrary languages, we need one circuit for each possible input length n.

Formal Definitions

- The **size** of a circuit is the number of *gates* in it.
- A **family of circuits** is an infinite sequence $\mathcal{C} = (C_0, C_1, \ldots)$ of boolean circuits, where C_n has n boolean inputs.
- $L \subseteq \{0,1\}^*$ has **polynomial circuits** if there is a family of circuits $\mathcal{C}$ such that:
  - The size of C_n is at most $p(n)$ for some fixed polynomial p.
  - For input $x \in \{0,1\}^*$, $C_{|x|}$ outputs 1 if and only if $x \in L$.
    * C_n accepts $L \cap \{0,1\}^n$.

The Circuit Complexity of P

Proposition 64 *All languages in P have polynomial circuits.*

- Let $L \in \mathrm{P}$ be decided by a TM in time $p(n)$.
- The construction in the proof of Theorem 25 (p. 169) gives, for any input of size n, a circuit with $O(p(n)^2)$ gates that accepts $L \cap \{0,1\}^n$.
- The size of the circuit depends only on L and the length of the input.
- The size of the circuit is polynomial in n.

Languages That Polynomial Circuits Accept

- It is untrue that polynomial circuits accept only languages in P.
- There are *undecidable* languages that have polynomial circuits.
  - Let $L \subseteq \{0, 1\}^*$ be an undecidable language.
  - Let $U = \{1^n :$ the binary expansion of n is in $L\}$.
  - U must be undecidable.
  - $U \cap \{1\}^n$ can be accepted by C_n that is trivially false if $1^n \notin U$ and trivially true if $1^n \in U$.
  - The family of circuits $(C_0, C_1, \ldots)$ is polynomial in size.

A Patch

- Despite their simplicity,
  - Circuits are *not* a realistic model of computation.
  - Polynomial circuits are *not* a plausible notion of efficient computation.
- What gives?
- The *effective and efficient constructibility* of $C_0, C_1, \ldots$.

Uniformity

- A family $(C_0, C_1, \ldots)$ of circuits is **uniform** if there is a $\log n$-space bounded TM which on input 1^n outputs C_n.
  - Circuits now cannot accept undecidable languages.
- A language has **uniformly polynomial circuits** if there is a *uniform* family of polynomial circuits that decides it.