Probabilistic Computing

Computation Complexity Seminar

Ali Fathi - Fahid Hamzei

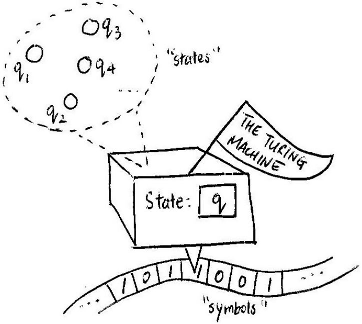
Department of Computer Science
Sharif University of Technology

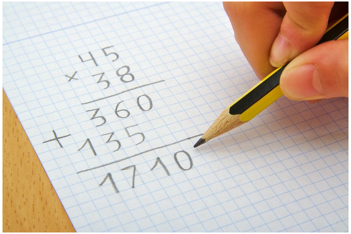
September 16, 2021

Overview

Introduction

Idea of Turing Machine





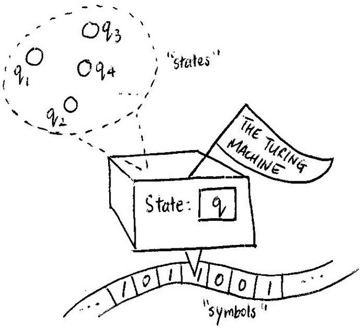
Intelligence, Uncertainty and Error

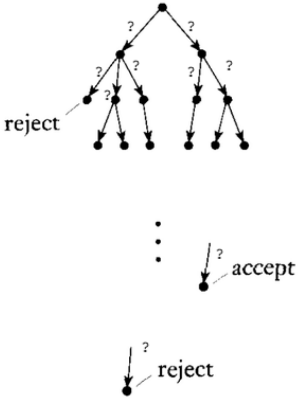
$$L = \{w_1, w_2, \cdots\}$$
$$w \in L \text{ or } w \notin L$$

Uncertainty and permission to make error, speed up the computations.

Power of Randomness

Probabilistic Turing Machine (Algorithm)





An assumption

- We access to a fair coin, or adequate number of random bites.
  (we will see other types of randomness, like unfair coin, fulfill our requirements too)

Example: Primality testing

$$PRIMES = \{p \in \mathbb{N} \,|\, p \text{ is prime}\}$$

Probabilistic Algorithm

On input p:
if p is even then
  if $p = 2$, accept and reject otherwise;
end
Sample k independent random numbers $a_1, \cdots, a_k \in \{1, \cdots, p-1\}$;
for $i = 1$ to k do
  Calculate $a_i^{p-1} \bmod p$. Reject if it is not 1;
  Factorize $p-1$ as $p-1 = s.2^l$ in which s is odd;
  Calculate series $y_0 = (a_i^{s.2^0} \bmod p), \cdots, y_l = (a_i^{s.2^l} \bmod p)$;
  If $y_0, \cdots, y_l$ wasn't like $(\cdots, -1, 1, 1, \cdots, 1)$, Reject;
end
If no reject occurred, accept;

Some lemmas

Lemma

If p is prime, then for all $a \in \{1, \cdots, p-1\}$, there is $a^{p-1} \bmod p = 1$.

For odd composite p:

- $\exists a \in \{1, \cdots, p-1\}$ such that $a^{p-1} \bmod p \neq 1$ (at least for half of such a's).
- $\forall a \in \{1, \cdots, p-1\}$: $a^{p-1} \bmod p = 1 \rightarrow$ Carmichael numbers.

Lemma

If p is a Carmichael numbers, $\exists q \notin \{1, -1\}$ such that $q^2 \bmod p = 1$.

Some lemmas

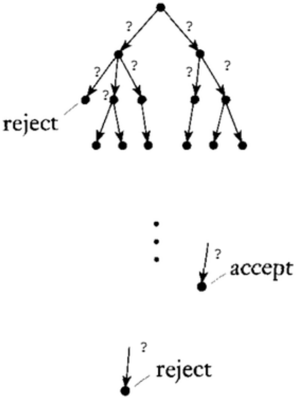
Lemma

For every odd composite p, for at least half of $\{1, \cdots, p-1\}$, the last non-one element of series $y_0, y_1, \cdots, y_l$ in which $y_t = (a^{s.2^t} \bmod p)$, is not -1.

Therefore are rejected with error less than 2^{-k}.

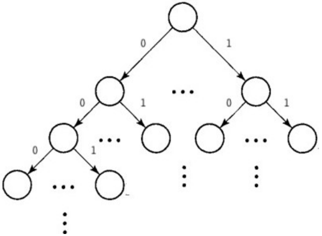
Probabilistic Turing Machine

Probabilistic Turing Machine (PTM)



Probabilistic Turing Machine

$$\langle Q, \Sigma, \Gamma, (\delta_0, \delta_1), q_0, q_a, q_r \rangle$$

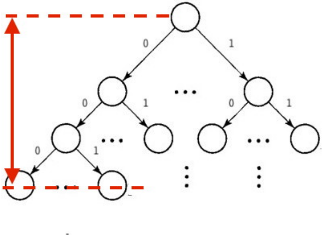


Tosses a coin at each node to use δ_0 or δ_1

Probabilistic Turing Machine

- $\delta_0 = \delta_1 \Rightarrow PTM$ would be a DTM
- $Pr[M \text{ accepts } x] = \sum_{b \text{ is accepting branch}} Pr[b] = \frac{\#\text{accepting branches}}{\#\text{all branches}}$ (considering same height for all branches)
- No non-determinism ($\Rightarrow$ practically implementable)

PTM computation time



Length of the longest computation branch

Bounded-error Probabilistic Decider (BP-Desider)

For a language L, PTM M is called a BP-Decider for L by error ϵ, if on every $x \in \Sigma^*$:

$$x \in L \Rightarrow Pr[M \text{ accepts } x] \geq 1 - \epsilon$$
$$x \notin L \Rightarrow Pr[M \text{ rejects } x] \geq 1 - \epsilon$$

i.e. $Pr[M(x) = L(x)] \geq 1 - \epsilon$.

Bounded-error Probabilistic TIME (BPTIME) classes

$$BPTIME(T(n)) = \{L \subseteq \Sigma^* \mid \exists M \in PTMs \text{ such that } \forall x : Pr[M(x) = L(x)] \geq \frac{2}{3}$$
$$\text{and } M \text{ decides } x \text{ on } O(T(|x|)) \text{ time}\}$$

We will see later that $\frac{2}{3}$ could be replaced with any number $p > \frac{1}{2}$.

Bounded-error Probabilistic Polynomial-time (BPP) class

$$BPP = \bigcup_{k=0}^{\infty} BPTIME(n^k)$$

BPP equivalent definition

$$BPP = \bigcup_{k=0}^{\infty} BPTIME(n^k)$$

Equivalently, $L \in BPP$ iff there exists some TM M (uses a bit stream r to decide about which transition function to chose) and a polynomial $P : \mathbb{N} \to \mathbb{N}$ which for every $x \in \Sigma^*$:

$$Pr_{r \in_R \{0,1\}^{P(|x|)}}[M(x,r) = L(x)] \geq \frac{2}{3}$$

BPP relationships

- $P \subseteq BPP$ (and is guessed $BPP = P$, though an open problem yet)
- $BPP \subseteq EXP$ (even $BPP \stackrel{?}{=} NEXP$ is open!)
- No proved relation with NP

Example

$$ZEROP = \{\langle p\rangle \,|\, p \text{ is a polynomial and } p(x_1, \cdots, x_n) \equiv 0\}$$

(Equivalent to polynomials equality, and same with RO branching program)

Probabilistic Algorithm

Lemma

if $p \not\equiv 0$, $deg(p) = d$ and $S \subseteq \mathbb{N}$ to be a finite set, for $a_1, \cdots, a_n$ randomly sampled from S:

$$Pr[p(a_1, \cdots, a_n) \neq 0] \geq 1 - \frac{d}{|S|}$$

For $d \leq 2^m$ and $S = \{1, \cdots, 10.2^m\}$, $1 - \frac{d}{|S|} \geq 0.9$.
$(m = |\langle p \rangle|$ and $\langle p \rangle$ is represented like a circuit with $\{\times, +, -\}$ instead of $\{\wedge, \vee, \neg\})$
Problem: terms like $a^d = O\left((10.2^m)^{2^m}\right)$.

Example (cont.)

Idea: do calculations in mod $k \in \{1, 2, \cdots, 2^{2m}\}$.

Lemma

Number y has at most $\log_2 y$ prime factors.

So for $y = O\left((10.2^m)^{2^m}\right)$, number of prime factors would be at most $(m + \log_2 10).2^m = o(\frac{2^{2m}}{2m}) < \frac{2^{2m}}{4m}$.

Lemma

There is at least $\frac{2^{2m}}{2m}$ prime numbers among $\{1, 2, \cdots, 2^{2m}\}$.

Therefore at least $\frac{2^{2m}}{4m}$ prime numbers, different to prime factors of y, exist in $\{1, 2, \cdots, 2^{2m}\}$. Hence $Pr[k \nmid y = p(a_1, \cdots, a_n)] \geq \frac{1}{4m}$.

RP, coRP and ZPP Classe

Randomized TIME (RTIME) classes

$RTIME(T(n))$ is the class of all languages like L for which there exists some PTM M such that for every input $x \in \Sigma^*$:

$$x \in L \Rightarrow Pr[M(x) = 1] \geq \frac{2}{3}$$
$$x \notin L \Rightarrow Pr[M(x) = 0] = 1$$

and running time of M is $O(T(|x|))$.
(Again $\frac{2}{3}$ could be replaced with any positive number $0 < p < 1$)
Note: we can be sure of result when we see $\underline{M(x) = 1}$.

Randomized Polynomial-time (RP) class

$$RP = \bigcup_{k=0}^{\infty} RTIME(n^k)$$

coRP class

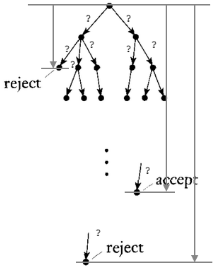
$$coRP = \{L \mid \bar{L} \in RP\}$$

RP and coRP relationships

- $RP \subseteq NP$
- $coRP \subseteq coNP$
- $RP, coRP \subseteq BPP$
- $P \subseteq RP, coRP$

Zero-sided error TIME (ZTIME) classes

$ZTIME(T(n))$ is the class of all languages like L for which there exists some PTM M such that for every input $x \in \Sigma^*$, $M(x) = L(x)$ and expected running time is $O(T(|x|))$.



Zero-sided error Probabilistic Polynomial-time (ZPP) class

$$ZPP = \bigcup_{k=0}^{\infty} ZTIME(n^k)$$

ZPP (other definition)

ZPP is the class of all languages like L for which there exists some polynomial-time PTM M such that for every input $x \in \Sigma^*$, $M(x) \in \{0, 1, \perp\}$ and $Pr[M(x) = \perp] < \frac{1}{2}$ and:

$$x \in L \Rightarrow M(x) \neq 0$$
$$x \notin L \Rightarrow M(x) \neq 1$$

Proof idea

- $ZPP_2 \subseteq ZPP_1$:
  Repeat M until anything else than $\perp$ appears.

$$\sigma = Pr[M(x) \neq \perp] \geq \frac{1}{2}$$

$$\mathbb{E}[T] = \sigma \cdot 1 + (1-\sigma)\left(\mathbb{E}[T]+1\right) \implies \mathbb{E}[T] = \frac{1}{\sigma} \leq 2$$

- $ZPP_1 \subseteq ZPP_2$:
  Halt after running $3T(n)$ and output $\perp$.

Markov Inequality

$Pr[T > a] \leq \frac{\mathbb{E}[T]}{a}$

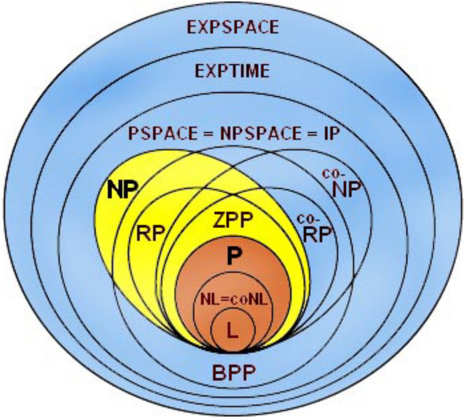
ZPP vs RP and coRP

Theorem

$ZPP = RP \cap coRP$

- $ZPP \subseteq RP \ (coRP)$:
  Halt after running $3T(n)$ and output 0 (1).
- $RP \cap coRP \subseteq ZPP$:
  M_1 no mistake on reject, and M_2 no mistake on accept.
  If $M_1(x) = M_2(x) = 1$, output 1; if $M_1(x) = M_2(x) = 0$, output 0; and otherwise $(M_1(x) = 0$ and $M_2(x) = 1)$ output $\perp$.

Overview


EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
co-
NP
RP
ZPP
co-
RP
P
NL=coNL
L
BPP

Any BPP-complete?

- Probably no! (hard to find any such language)
- Semantic definition of $BP - Decider$: at least $\frac{2}{3}$ branches of M must be equal (checking this characteristic is undecidable).
- Example:
  $L = \{\langle M, x, 1^t\rangle \mid M$ accepts x with probability more than $\frac{2}{3}$ in at most t steps$\}$
  U_{BPP}: simulator $\Rightarrow L \in BPP - hard$
  but U_{BPP} does not hold the characteristic for ill M's.

Any hierarchy theorem for BPP?

- Probably not and again hard
- Even unknown: $BPTIME(n) \stackrel{?}{=} BPTIME(n^{(\log n)^{10}})$

Robustness

Robustness

$$L \in \mathrm{BPP} \iff \begin{cases} Pr[M(x) = L(x)] \geq \frac{2}{3} \\ \\ Pr[M(x) \neq L(x)] \leq \frac{1}{3} \end{cases}$$

What's so special about $\frac{1}{3}$?

The magnitude of failure

$$L \in \mathrm{BPP} \iff \begin{cases} Pr[M(x) = L(x)] \geq \frac{2}{3} \\ \\ Pr[M(x) \neq L(x)] \leq \frac{1}{3} \end{cases}$$

What's so special about $\frac{1}{3}$? Nothing! any value less than $\frac{1}{2}$ works.

The magnitude of failure

$\forall\, c > 0:$

$BPP_{\frac{1}{2}+n^{-c}} := \{L \mid \exists \text{ poly-time } PTM\ \ M,\ \forall\, x:\ Pr[M(x) = L(x)] \geq \frac{1}{2} + |x|^{-c}\}$

The magnitude of failure

$\forall\, c > 0:$
$BPP_{\frac{1}{2}+n^{-c}} := \{L \mid \exists \text{ poly-time } PTM\ \ M,\ \forall\, x:\ Pr[M(x) = L(x)] \geq \frac{1}{2} + |x|^{-c}\}$

- n^{-c} is noticeable
- success probability $\geq \frac{1}{2} + n^{-c}$
- $BPP = BPP_{\frac{2}{3}}$

The magnitude of failure

$\forall\, c > 0:$

$BPP_{\frac{1}{2}+n^{-c}} := \{L \mid \exists \text{ poly-time } PTM \;\; M,\; \forall\, x:\; Pr[M(x) = L(x)] \geq \frac{1}{2} + |x|^{-c}\}$

- n^{-c} is noticeable
- success probability $\geq \frac{1}{2} + n^{-c}$
- $BPP = BPP_{\frac{2}{3}}$

$BPP_{\frac{2}{3}} \stackrel{?}{=} BPP_{\frac{3}{4}} \stackrel{?}{=} BPP_{\frac{501}{1000}}$

Amplification lemma

Theorem (Error reduction for BPP)

$\forall\, c > 0,\ \exists$ poly-time $PTM\ \ M,\ \forall x : Pr[M\,(x) = L\,(x)] \geq \frac{1}{2} + |x|^{-c}$

$$\Downarrow$$

$\forall\, d > 0,\ \exists$ poly-time $PTM\ \ M',\ \forall x : Pr[M'\,(x) = L\,(x)] \geq 1 - 2^{-|x|^{d}}$

Amplification lemma

Theorem (Error reduction for BPP)

$\forall\, c > 0,\ \exists$ poly-time $PTM\ \ M,\ \forall x : Pr[M\,(x) = L\,(x)] \geq \frac{1}{2} + |x|^{-c}$

$$\Downarrow$$

$\forall\, d > 0,\ \exists$ poly-time $PTM\ \ M',\ \forall x : Pr[M'\,(x) = L\,(x)] \geq 1 - 2^{-|x|^d}$

- error probability $\leq 2^{-|x|^d}$ (negligible)

Amplification lemma

Theorem (Error reduction for BPP)

$\forall\, c > 0,\ \exists$ poly-time $PTM\ \ M,\ \forall x : Pr[M\,(x) = L\,(x)] \geq \frac{1}{2} + |x|^{-c}$

$$\Downarrow$$

$\forall\, d > 0,\ \exists$ poly-time $PTM\ \ M',\ \forall x : Pr[M'\,(x) = L\,(x)] \geq 1 - 2^{-|x|^d}$

Proof idea :

- M' simulates M many times
- Output the majority value

Error reduction Algorithm

M' : on input x

1. Simulate M on x for $8|x|^{2c+d}$ times.
2. If the majority of outputs are 1, *output* 1; otherwise *output* 0.

$$BPP_{\frac{1}{2}+n^{-c}} = BPP$$

Theorem

$\forall\ c > 0 : BPP_{\frac{1}{2}+n^{-c}} = BPP$

$BPP_{\frac{1}{2}+n^{-c}} = BPP$

Theorem

$\forall\ c > 0 : BPP_{\frac{1}{2}+n^{-c}} = BPP$

- Proof : Clearly $BPP = BPP_{\frac{2}{3}} \subseteq BPP_{\frac{1}{2}+n^{-c}}$

$$BPP_{\frac{1}{2}+n^{-c}} \subseteq BPP_{1-2^{-n^d}} \subseteq BPP_{\frac{2}{3}} = BPP$$

Unfair Coin

- Fair coin : $Pr[Head] = Pr[Tail] = \frac{1}{2}$
- ρ-biased coin : $\begin{cases} Pr[Head] = \rho \\ Pr[Tail] = 1 - \rho \end{cases} \quad 0 \leq \rho \leq 1$
- Turing Machine + Fair coin VS. Turing Machine + Unfair coin

Unfair Coin

Theorem

- ρ-biased coin : $\begin{cases} Pr[Head] = \rho \\ Pr[Tail] = 1 - \rho \end{cases}$
- The ith bit of ρ is computable in $poly\,(i)$ time

$\Longrightarrow$ ρ-biased coin can be simulated with a standard PTM in expected time $O\,(1)$

Unfair Coin

P : on input $\rho = 0.\rho_1\rho_2\cdots$

For $i = 1, 2, \cdots$:

Generate random bit b_i

1. If $b_i > \rho_i$, output $Tail$.
2. If $b_i < \rho_i$, output $Head$.

Unfair Coin

P : on input $\rho = 0.\rho_1\rho_2\cdots$

For $i = 1, 2, \cdots$:

Generate random bit b_i

1. If $b_i > \rho_i$, output $Tail$.
2. If $b_i < \rho_i$, output $Head$.

example : let $\rho = \frac{3}{4} = (0.11000\cdots)_2$

$$\begin{cases} 0 \to Head & (\frac{1}{2}) \\ 1 \begin{cases} 0 \to Head & (\frac{1}{4}) \\ 1 \begin{cases} 0 \to \cdots Tail & (\frac{1}{8}) \\ 1 \to Tail & (\frac{1}{8}) \end{cases} \end{cases} \end{cases}$$

Unfair Coin

P : on input $\rho = 0.\rho_1\rho_2\cdots$
For $i = 1, 2, \cdots$:
Generate random bit b_i
1. If $b_i > \rho_i$, output $Tail$.
2. If $b_i < \rho_i$, output $Head$.

$\rho = \rho_1 \cdot 2^{-1} + \rho_2 \cdot 2^{-2} + \cdots$

$Pr[Head] = \sum_{i=1}^{\infty} \rho_i \frac{1}{2^i} = \rho$

$E[T] = \sum_{i=1}^{\infty} E[T \mid i_{stop} = i] \cdot Pr[i_{stop} = i] = \sum_{i=1}^{\infty} i^c \frac{1}{2^i} = O(1)$

Unfair Coin

Theorem

- Fair coin:$Pr[Head] = Pr[Tail] = \frac{1}{2}$
- $PTM\ \ M$ has access to a ρ-biased coin.

$\Longrightarrow$ Fair coin can be simulated by M in expected time $O\left(\frac{1}{\rho(1-\rho)}\right)$

Unfair Coin

1. Generate two bits b_1 and b_2
2. If $b_1 = Head$ and $b_2 = Tail$, output $Head$
3. If $b_1 = Tail$ and $b_2 = Head$, output $Tail$
4. If $b_1 = b_2$ Generate two fresh random bits then go to 2.

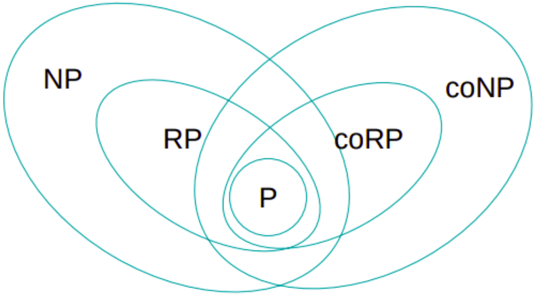
Unfair Coin

1. Generate two bits b_1 and b_2
2. If $b_1 = Head$ and $b_2 = Tail$, output $Head$
3. If $b_1 = Tail$ and $b_2 = Head$, output $Tail$
4. If $b_1 = b_2$ Generate two fresh random bits then go to 2.

$Pr[Halt]$ in each step $= \rho\,(1-\rho) \implies E[T] = \frac{1}{\rho(1-\rho)} = O\,(1)$

BPP

- We know :



BPP is in PH

Theorem

$BPP \subseteq \Sigma_2^p \cap \Pi_2^p$

Adleman's theorem

Theorem (Adleman's theorem)

$BPP \subset P_{/poly}$

Adleman's theorem

Theorem (Adleman's theorem)

$BPP \subset P_{/poly}$

- Goal : $\exists\, r_0, M \;, \forall x : |x| = n \implies M(x, r_0) = L(x)$
- Proof idea : $\begin{cases} \text{error reduction} \\ \text{union bound} \end{cases}$

Adleman's theorem

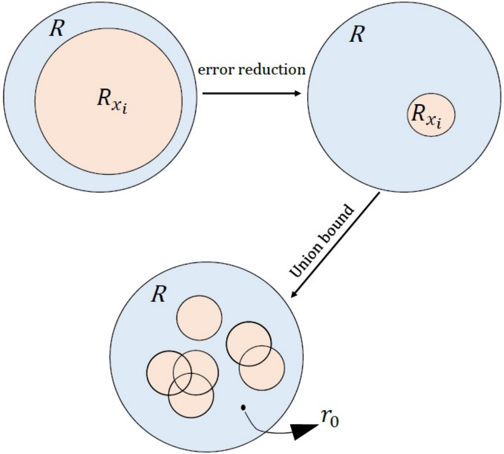
$$|x| = n \text{ and } |r| = m$$

$$R_{x_i} = \{r \mid M(x, r) \neq L(x)\}$$

$$\Pr_{r \in_R \{0,1\}^m} [M(x_i, r) \neq L(x_i)] < \epsilon$$

$$\Longrightarrow \frac{|R_{x_i}|}{|R|} < \epsilon \Longrightarrow |R_{x_i}| < \epsilon |R|$$

$$\sum_{i=1}^{2^n} |R_{x_i}| < |R|$$



Randomized Reduction

Randomized Reduction

Definition (Randomized reduction)

$B \leq_r C := \exists$ polynomial-time $PTM\ M, \forall x : Pr[C(M(x)) = B(x)] \geq \frac{2}{3}$

Randomized Reduction

Definition (Randomized reduction)

$B \leq_r C := \exists\,\text{polynomial-time}\, PTM\;\; M, \forall\, x : Pr[C\,(M\,(x)) = B\,(x)] \geq \frac{2}{3}$

- $B \leq_p C := \exists\, M, \forall\, x : Pr[C\,(M\,(x)) = B\,(x)] = 1$
- Not transitive
- $M\,(x) \in \{0, 1\}^*$

Randomized Reduction

Definition (Randomized reduction)

$B \leq_r C := \exists \text{polynomial-time}\, PTM\ \ M, \forall\, x : Pr[C\,(M\,(x)) = B\,(x)] \geq \frac{2}{3}$

- $B \leq_p C := \exists\, M, \forall\, x : Pr[C\,(M\,(x)) = B\,(x)] = 1$
- Not transitive
- $M\,(x) \in \{0, 1\}^*$

Definition (transitive randomized reduction)

$B \leq_r C :=$
$\exists \text{polynomial-time}\, PTM\ \ M, \forall\, x : Pr[C\,(M\,(x)) = B\,(x)] \geq 1 - 2^{-|x|^d}$

Randomized Reduction

Observation : $C \in BPP \ \& \ B \leq_r C \implies B \in BPP$

Randomized Reduction

Observation : $C \in BPP \;\&\; B \leq_r C \implies B \in BPP$

$\exists\, M_C, \forall\, x : Pr[M_C\,(x) = C\,(x)] \geq \frac{2}{3} \;\rightarrow\; Pr[M'_C\,(x) = C\,(x)] \geq \frac{11}{12}$

$\exists\, M_r, \forall\, x : Pr[C\,(M_r\,(x)) = B\,(x)] \geq \frac{2}{3}$

M_B : On input x

1. Simulate M_r on x to obtain $M_r\,(x)$
2. Simulate M'_c on $M_r\,(x)$, output $M'_c\,(M_r\,(x))$

$BP \cdot \mathcal{C}$

Definition : $BP \cdot \mathcal{C}$

$\forall \mathcal{C}$: (Class of languages)

$L \in BP \cdot \mathcal{C} \iff \exists\, D \in \mathcal{C}$ & polynomial-time $PTM\ M$:

$$\begin{cases} x \in L \implies Pr[M(x) \in D] \geq \frac{2}{3} \\ x \notin L \implies Pr[M(x) \notin D] \geq \frac{2}{3} \end{cases}$$

$BP \cdot \mathcal{C}$

Definition : $BP \cdot \mathcal{C}$

$\forall \mathcal{C}$: (Class of languages)

$L \in BP \cdot \mathcal{C} \iff \exists\, D \in \mathcal{C}$ & polynomial-time $PTM\ M$:

$$\begin{cases} x \in L \implies Pr[M(x) \in D] \geq \frac{2}{3} \\ x \notin L \implies Pr[M(x) \notin D] \geq \frac{2}{3} \end{cases}$$

Theorem

$BP \cdot NP = \{L \mid L \leq_r SAT\}$

- $NP = \{L \mid L \leq_p SAT\}$

$BP \cdot NP$(Probabilistic NP)

Theorem

$BP \cdot NP = \{L \,|\, L \leq_r SAT\}$

$\{L \,|\, L \leq_r SAT\} \subseteq BP \cdot NP$ by definition.

$BP \cdot NP \subseteq \{L \,|\, L \leq_r SAT\}$:
$C \in BP \cdot NP \implies \exists\, D \in NP : C \leq_r D$

$$\left.\begin{array}{l} C \leq_r D \\ D \leq_p SAT \end{array}\right\} C \leq_r D \leq_p SAT \implies C \leq_r SAT$$

$BP \cdot NP$(Probabilistic NP)

Theorem

$BP \cdot NP = \{L \,|\, L \leq_r SAT\}$

$\{L \,|\, L \leq_r SAT\} \subseteq BP \cdot NP$ by definition.

$BP \cdot NP \subseteq \{L \,|\, L \leq_r SAT\}$:
$C \in BP \cdot NP \implies \exists\, D \in NP : C \leq_r D$

$$\left.\begin{array}{l} C \leq_r D \\ D \leq_p SAT \end{array}\right\} C \leq_r D \leq_p SAT \implies C \leq_r SAT$$

- $BPP \subseteq BP \cdot NP$
- $NP \subseteq BP \cdot NP$

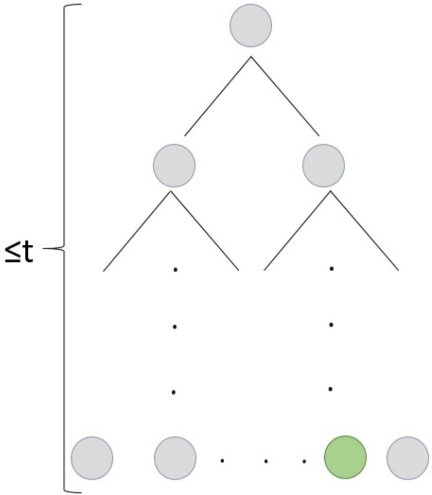
Randomized Space-Bounded Computation

$$L \in RL \iff \exists\ O\left(\log n\right)\text{-space }\ PTM\ M : \begin{cases} x \in L \implies Pr[M\left(x\right) = 1] \geq \frac{2}{3} \\ x \notin L \implies Pr[M\left(x\right) = 0] = 1 \end{cases}$$

$$L \in BPL \iff \exists\ O\left(\log n\right)\text{-space }\ PTM\ M : \begin{cases} x \in L \implies Pr[M\left(x\right) = 1] \geq \frac{2}{3} \\ x \notin L \implies Pr[M\left(x\right) = 0] \geq \frac{2}{3} \end{cases}$$

Definitional Issue

$RL \subseteq NL$ similar to $RP \subseteq NP$



Definitional Issue

$NL \subseteq RL:$

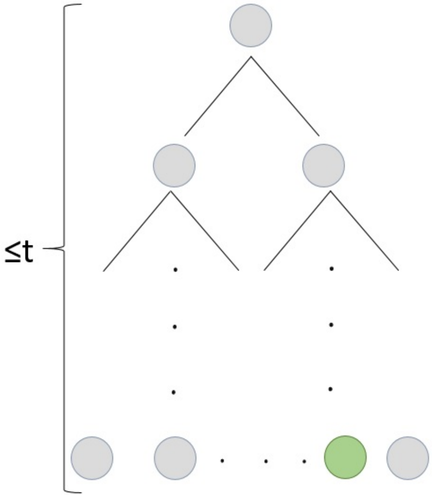
$L \in NL \implies \exists \text{poly-time}\, N : N(x) = L(x)$

$t = P(|x|)$

$Pr[\text{find an accepting path}] \geq 2^{-t}$

$E[T] = 2^t$

$O(\log \log n)$ counter



UPATH

- $UPATH$: contains all $\langle G, s, t \rangle$:
  - G is an undirected graph
  - s and t are connected in G
- Recall $PATH$ is NL-complete
- Actually $UPATH \in L$

UPATH

- $UPATH$: contains all $\langle G, s, t \rangle$:
  - G is an undirected graph
  - s and t are connected in G
- Recall $PATH$ is NL-complete
- Actually $UPATH \in L$

M : On input $\langle G, s, t \rangle$

1. Take a random walk of length $100n^4$ starting from s
2. Accept iff the walk reaches t within $100n^4$ steps.

References

References

- Computational complexity, a modern approach; Sanjeev Arora, Boak Barak
- Introduction to the theory of computation; Michael Sipser; Third edition
- Computational complexity, a conceptual perspective; Oded Goldreich
- Rafael Pass's lecture notes on Theory of Computing
- Markov Chains and Random Walks

Thanks :))