I

Klasy zrandomizowane

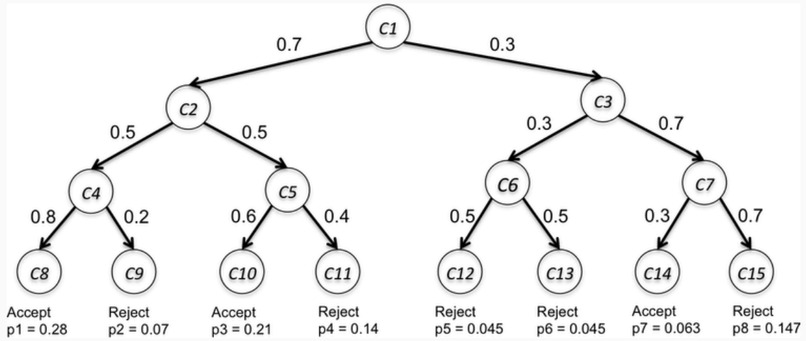
23 stycznia 2025 r.

MATEUSZ BUDZYŃSKI
MACIEJ CIEPIELA

Spis Treści

III

Probabilistyczna Maszyna Turinga



Taką maszynę można zdefiniować jako $M = (Q, \Sigma, \Gamma, q_0, A, \delta_1, \delta_2)$
gdzie:

- Q to skończona ilość stanów
- Σ to alfabet na wejściu
- Γ to alfabet taśmy, w tym blank - #
- $q_0 \in Q$ to stan początkowy
- $A \subseteq Q$ to zbiór stanów akceptujących
- $\delta_1 : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ to pierwsza probabilistyczna funkcja przejścia. L to ruch w lewo na taśmie, a R w prawo
- $\delta_2 : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ to druga probabilistyczna

PTM CD.

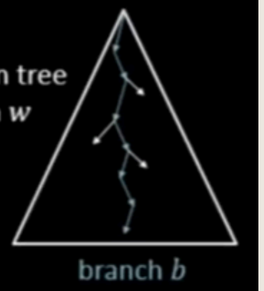
IV

Probabilistic TMs

Defn: A probabilistic Turing machine (PTM) is a variant of a NTM where each computation step has 1 or 2 possible choices.

$\Pr[\text{ branch } b\text{ }] = 2^{-k}$ where b has k coin flips

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



$\Pr[\text{ } M \text{ rejects } w\text{ }] = 1 - \Pr[\text{ } M \text{ accepts } w\text{ }]$

Defn: For $\epsilon \geq 0$ say PTM M decides language A with error probability ϵ if for every w, $\Pr[\text{ } M \text{ gives the wrong answer about } w \in A\text{ }] \leq \epsilon$

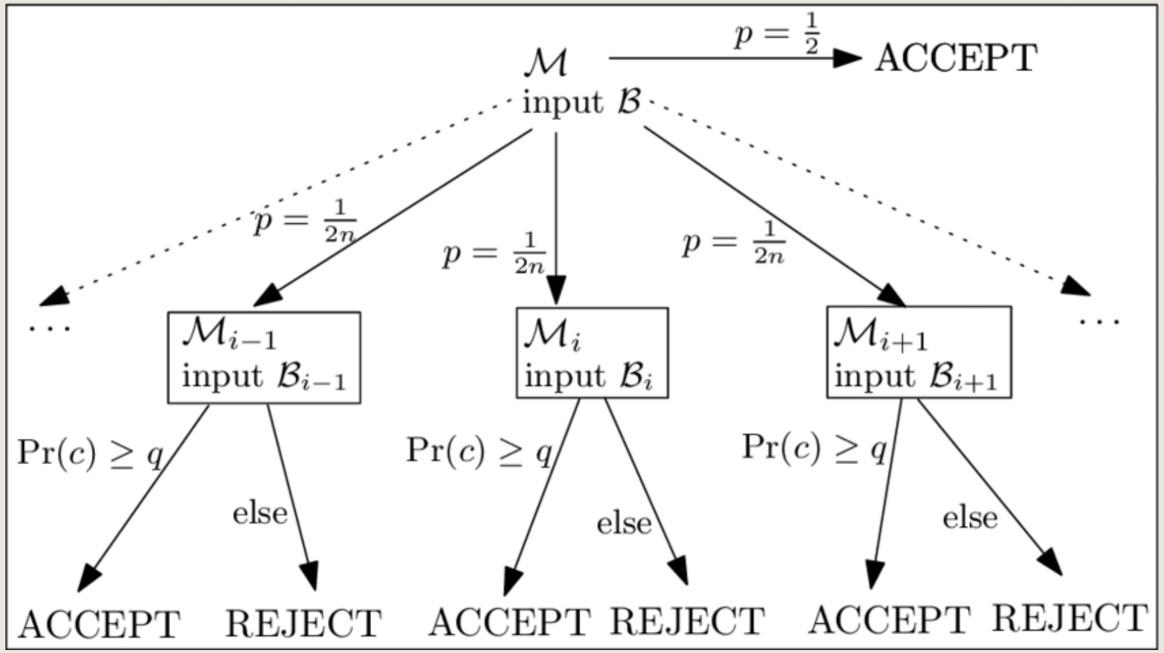
i.e., $w \in A \rightarrow \Pr[\text{ } M \text{ rejects } w\text{ }] \leq \epsilon$

$w \notin A \rightarrow \Pr[\text{ } M \text{ accepts } w\text{ }] \leq \epsilon$.

For a PTM M, and input x, we define the random variable $T_{M,x}$ to be the running time of M on input x. That is, $Pr[T_{M,x} = T] = p$ if with probability p over the random choices of M on input x, it will halt within T steps.

We say that M has an *expected running time* $T(n)$ if the expectation $E[T_{M,x}]$ is at most $T(|x|)$ for every $x \in \{0,1\}^*$.

NIE MUSIMY OGRANICZAĆ SIĘ TYLKO DO DWÓCH ROZGAŁĘZIEŃ, ORAZ DO PRAWDOPODOBIEŃSTWA "PRZEJŚCIA" = 1/2 ALE DO ANALIZY ZE WZGLĘDU NA PROSTOTĘ KORZYSTA SIĘ Z TAKIEGO MODELE. POZOSTAŁE MOŻNA W PROSTY SPOSÓB DO NIEGO SPROWADZIĆ.



Przykład działania (7.2.1)

We sketch an algorithm showing that PRIMES is in **BPP** (and in fact in **coRP**). For every number N, and $A \in [N-1]$, define

$$QR_N(A) = \begin{cases} 0 & gcd(A,N) \neq 1 \\ +1 & A \text{ is a } \textit{quadratic residue} \text{ modulo } N \\ & \text{That is, } A = B^2 \pmod N \text{ for some } B \text{ with } gcd(B,N) = 1 \\ -1 & \text{otherwise} \end{cases}$$

We use the following facts that can be proven using elementary number theory:

- For every odd prime N and $A \in [N-1]$, $QR_N(A) = A^{(N-1)/2} \pmod N$.
- For every odd N, A define the *Jacobi symbol* $(\frac{N}{A})$ as $\prod_{i=1}^{k} QR_{P_i}(A)$ where $P_1, \ldots, P_k$ are all the (not necessarily distinct) prime factors of N (i.e., $N = \prod_{i=1}^{k} P_i$). Then, the Jacobi symbol is computable in time $O(\log A \cdot \log N)$.
- For every odd composite N, $\left|\{A \in [N-1] : gcd(N,A) = 1 \text{ and } (\frac{N}{A}) = A^{(N-1)/2}\}\right| \leq \frac{1}{2}\left|\{A \in [N-1] : gcd(N,A) = 1\}\right|$

Together these facts imply a simple algorithm for testing primality of N (which we can assume without loss of generality is odd): choose a random $1 \leq A < N$, if $gcd(N,A) > 1$ or $(\frac{N}{A}) \neq A^{(N-1)/2} \pmod N$ then output "composite", otherwise output "prime". This algorithm will always output "prime" is N is prime, but if N is composite will output "composite" with probability at least $1/2$. (Of course this probability can be amplified by repeating the test a constant number of times.)

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & \text{if } a \equiv 0 \pmod p, \\ 1 & \text{if } a \not\equiv 0 \pmod p \text{ and for some integer } x\text{: } a \equiv x^2 \pmod p, \\ -1 & \text{if } a \not\equiv 0 \pmod p \text{ and there is no such } x. \end{cases}$$

PRIME(A, N):
  IF $(gcd(A,N) > 1)$ or $((N/A) \neq A^{(N-1)/2} \; mod \; N)$
    output COMPOSITE
  ELSE
    output PRIME

Theorem

PRIMES $\in$ *BPP*

- Jeśli N jest złożona, to $A^{(N-1)/2} \mod N$ różni się od $\left(\frac{A}{N}\right)$ dla co najmniej połowy liczb A.
- Dowodzi się tego, analizując strukturę reszt modulo N i faktoryzację N, co prowadzi do ograniczenia oszukujących wartości A.

VI

Algorytm losowy

Algorytm losowy to algorytm, w którym dozwolony jest dostęp do niezależnych, niezaburzonych i losowych bitów. taki algorytm wtedy może wykorzystać te zrandomizowane bity do wpłynięcia na przebieg obliczeń.

Klasy RP, coRP

Algorytmy RP i coRP mają jednostronne prawdopodobieństwo błędu i są często nazywane algorytmami Monte Carlo

Definiujemy klase RTIME(t(n)), która zawiera w sobie każdy język *L*, dla którego istnieje PTM *M*, działająca w czasie *t(n)*, taka że:

$$x \in L \Rightarrow \mathbf{Pr}[M \text{ accepts } x] \geq \frac{2}{3}$$
$$x \notin L \Rightarrow \mathbf{Pr}[M \text{ accepts } x] = 0$$

RP i coRP definiujemy jako:

$$\mathbf{RP} = \cup_{c>0}\mathbf{RTIME}(n^c)$$

$$\mathbf{coRP} = \{L \mid \overline{L} \in \mathbf{RP}\}$$

Theorem

PRIMES $\in$ *coRP*

$$\mathsf{P} \stackrel{?}{=} \mathsf{RP}$$

VIII

Klasa ZPP

Algorytmy ZPP mają zerowe prawdopodobieństwo błędu i są często nazywane algorytmami Las Vegas

Algorytm Las Vegas dla L można również postrzegać jako taki, który zapewnia trzy możliwe odpowiedzi: „akceptuj”, „odrzuć” i „nie wiem”. Nigdy nie akceptuje ciągu spoza L i nigdy nie odrzuca ciągu wewnątrz L. Wynik „nie wiem” ma prawdopodobieństwo 1/2.

Klasa ZTIME(T(n)), zawiera w sobie każdy język L, dla którego istnieje maszyna z oczekiwanym czasem O(T(n)), która nigdy nie popełnia błędu, tj.:

$$x \in L \Rightarrow \mathbf{Pr}[M \text{ accepts } x] = 1$$
$$x \notin L \Rightarrow \mathbf{Pr}[M \text{ halts without accepting on } x] = 1$$

ZPP definiujemy jako:

$$\mathbf{ZPP} = \cup_{c>0}\mathbf{ZTIME}(n^c)$$

ZPP = RP ∩ coRP

Klasa BPP

Defn: BPP $= \{A|$ some poly-time PTM decides A with error $\epsilon = 1/3\}$

Amplification lemma: If M_1 is a poly-time PTM with error $\epsilon_1 < 1/2$ then, for any $0 < \epsilon_2 < 1/2$, there is an equivalent poly-time PTM M_2 with error ϵ_2. Can strengthen to make $\epsilon_2 < 2^{\text{poly}(n)}$.

Proof idea: $M_2 =$ "On input w
1. Run M_1 on w for k times and output the majority response."

Details: Calculation to obtain k and the improved error probability.

Significance: Can make the error probability so small it is negligible.

The choice of the constant $2/3$ seemed pretty arbitrary. We now show that we can replace $2/3$ with any constant larger than $1/2$ and in fact even with $1/2 + n^{-c}$ for a constant $c > 0$.

DOWÓD COMPUTATIONAL COMPLEXITY: A MODERN APPROACH - ARORA, BARAK ROZDZIAŁ 7.4.1

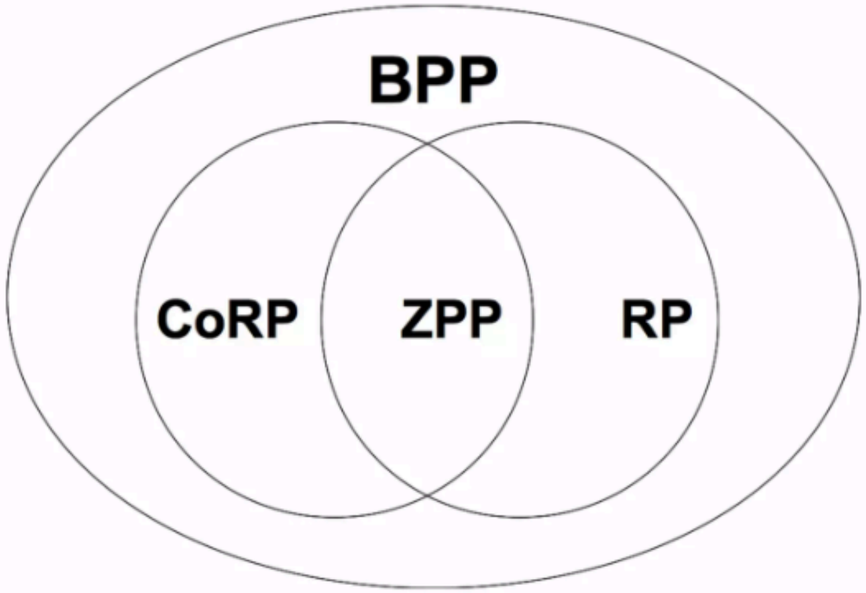
BPP algorithm (1 run)		
Correct answer \ Answer produced	**Yes**	**No**
Yes	≥ 2/3	≤ 1/3
No	≤ 1/3	≥ 2/3

BPP algorithm (k runs)		
Correct answer \ Answer produced	**Yes**	**No**
Yes	$> 1 - 2^{-ck}$	$< 2^{-ck}$
No	$< 2^{-ck}$	$> 1 - 2^{-ck}$
for some constant $c > 0$		

BPP = P ?

X

BPP a inne klasy



- Klasa **BPP** jest najbardziej ogólna spośród tych trzech klas, ponieważ:
  - Pozwala na błędy dwustronne,
  - Ma wyraźnie określone ograniczenie na margines błędu ($\leq \frac{1}{3}$).
- Klasy **RP** i **ZPP** są bardziej restrykcyjne:
  - RP ogranicza błędy do jednostronnych,
  - ZPP nie dopuszcza żadnych błędów, ale może potrzebować więcej czasu (w oczekiwaniu).

Theorem

$P \subseteq BPP \subseteq EXP.$

Proof.

- Since a deterministic TM is a special case of a PTM (where both transition functions are equal) BPP clearly contains P.
- Given a polynomial-time PTM M and input $x \in \{0,1\}^n$ in time $2^{poly(n)}$ it is possible to enumerate all possible random choices and compute precisely the probability that $M(x) = 1$.

(NA TABLICY)

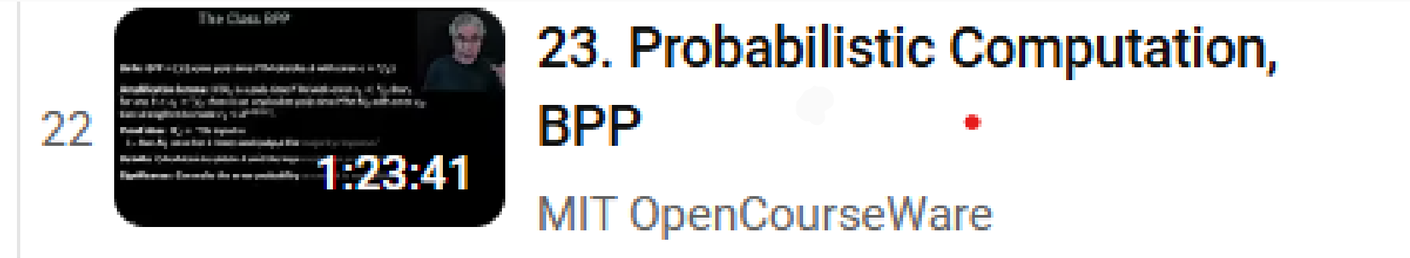
□

1. $P \subseteq BPP$
2. $BPP \subseteq EXP$
3. $RP \subseteq BPP$
4. $coRP \subseteq BPP$
5. $ZPP = RP \cap coRP$
6. $ZPP \subseteq BPP$

XI

Źródła

- Computational Complexity: A Modern Approach - Arora, Barak
- https://www.slideshare.net/slideshow/randomized-computation/23217936#5
- Wikipedia
- https://cse.iitkgp.ac.in/~abhij/course/theory/CC/Spring04/chap5.pdf
-



XII

Dziękujemy za uwagę!