Algorithms and Computation

(grad course)

Lecture 11: Randomized Algorithms II

Instructor: Hossein Jowhari

jowhari@kntu.ac.ir

Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

Fall 2021

Problems with one-sided error algorithms:

- **RP** (Randomized Polynomial time)
- **coRP** (Complement of Randomized Polynomial time)

Problems with zero-error algorithms:

- **ZPP** (Zero-error Polynomial time)

Problems with two-sided error algorithms:

- **PP** (Probabilistic Polynomial time)
- **BPP** (Bounded-error Probabilistic Polynomial time)

One-sided Error Algorithms

Definition: The class **RP** (Randomized Polynomial time) consists of all languages L that have a randomized algorithm A that runs in polynomial time such that for any input $x \in \Sigma^*$

- $x \in L \Rightarrow Pr[A(x) \text{ accepts}] \geq \frac{1}{2}$
- $x \notin L \Rightarrow Pr[A(x) \text{ accepts}] = 0$

Definition: The class **coRP** consists of all languages L that have a randomized algorithm A that runs in polynomial time such that for any input $x \in \Sigma^*$

- $x \in L \Rightarrow Pr[A(x) \text{ accepts}] = 1$
- $x \notin L \Rightarrow Pr[A(x) \text{ accepts}] \leq \frac{1}{2}$

One-sided error algorithms: Examples

- Verifying polynomial identity
- Verifying matrix multiplication

$$A \times B = C?$$

  Choose random vextor x and check if $ABx = Cx$

- Global min-cut in graphs

Zero-error randomized algorithms

Definition: The language L belongs to the class **ZPP** if and only if there is a randomized algorithm A that given $x \in \Sigma^*$ gives the correct answer on the question whether $x \in L$ or not. The running time of the algorithm A is polynomial in expectation.

In some literature, algorithm A is called a Las Vegas algorithm. Monte Carlo algorithms are randomized algorithms with non-zero error probability.

Expectation

Definition: Let X be a discrete random variable taking values from the domain U. The expectation of X defined by

$$E[X] = \sum_{u \in U} u Pr[X = u]$$

Example: Let X be the outcome of rolling a dice. We have

$$E[X] = 1\frac{1}{6} + 2\frac{1}{6} + 3\frac{1}{6} + 4\frac{1}{6} + 5\frac{1}{6} + 6\frac{1}{6} = 3.5$$

Expected Running time

Expected running time of algorithm A on input x:

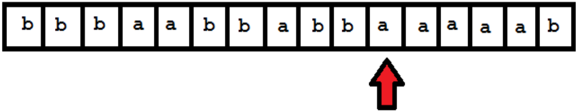
$$E[\text{running time}(A, x)] =$$

$$\sum_{\text{A stops on } x \text{ and } r_1,\ldots,r_t} = \frac{1}{2^t} \times \text{running time}(A, x, r_1, \ldots, r_t)$$

Expected running time of algorithm A on input of length n:

$$E[\text{running time}(A)] = \max_{x,|x|=n} \{E[\text{running time}(A, x)]\}$$

Example: Given an array A of length n containing $n/2$ number of a's and $n/2$ number of b's, find the position of an a.



Randomized Algorithm: Randomly pick a position and check if it is an a. Repeat this until an a is found.

Analysis: $\Pr[\text{failure}] = 0$. The algorithm succeeds with probability 1.

$$\text{Expected running time} = \sum_{i}^{\infty} \frac{i}{2^i} = 2$$

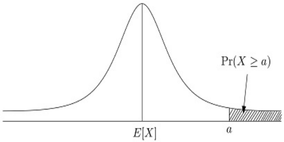
- **Linearity of expectation.** $E[X+Y] = E[X] + E[Y]$
- **Markov Inequality.** Given random variable X that takes <u>positive</u> values, for $a > 0$

$$Pr[X \geq a] \leq \frac{E[X]}{a}$$

Setting $a = tE[X]$ where $t > 0$, we get

$$Pr[X \geq tE[X]] \geq \frac{1}{t}$$

$$\begin{aligned}
E[X] &= \sum_x xP(x) \\
&= \sum_{x<a} xP(x) + \sum_{x \geq a} xP(x) \\
&\geq \sum_{x \geq a} xP(x) \\
&\geq \sum_{x \geq a} aP(x) \\
&= a \sum_{x \geq a} P(x) \\
&= aP(x \geq a),
\end{aligned}$$



Example: Waiting for a first success

Suppose we have a biased coin that comes up HEAD with probability p.

What is the expected number of coin tosses until a HEAD comes up?

Let X be the random variable equal to the number of tosses performed.

For $j > 0$, we have $Pr[X = j] = (1 - p)^{j-1}p$

$E[X] = \sum_{j=1}^{\infty} j.Pr[X = j] = \sum_{j=1}^{\infty} j(1 - p)^{j-1}p = \frac{1}{p}$

Theorem: $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$.

Lemma: $\mathbf{ZPP} \subseteq \mathbf{RP} \cap \mathbf{coRP}$.

Proof: Let L be a decision problem in $\mathbf{ZPP}$. There is a randomized algorithm A that decides $x \in L$. Let $f(n)$ be the expected running time of A on an input of size n. $f(n)$ is a polynomial function.

We want to show $L \in \mathbf{RP} \cap \mathbf{coRP}$. In other words:

- There is a randomized algorithm A_1 with polynomial time complexity where $x \in L \Rightarrow Pr[A_1(x) \text{ accepts}] \geq \frac{1}{2}$ and $x \notin L \Rightarrow Pr[A_1(x) \text{ accepts}] = 0$
- There is a randomized algorithm A_2 with polynomial time complexity where $x \in L \Rightarrow Pr[A_2(x) \text{ accepts}] = 1$ and $x \notin L \Rightarrow Pr[A(x) \text{ accepts}] \leq \frac{1}{2}$

Algorithm A_1: Run the zero-error algorithm A on input x. Stop the execution before the running time exceeds $2f(n)$.

- If A accepts x before it is stopped, we accept x.
- If A rejects x before it is stopped, we reject x.
- Otherwise (if A is stopped before giving an answer) we accept x.

If $x \in L$, algorithm A_1 always accepts x.

If $x \notin L$, algorithm A_1 accepts x with probability at most $\frac{1}{2}$. This happens when the running time of A on x exceeds $2f(n)$. Let E be this event.

Let R be the running time of A on x. Since $E[R] \leq f(n)$

$$Pr[E] = Pr[R \geq 2f(n)] \leq \frac{1}{2} \qquad \text{(Markov inequality)}$$

Therefore $L \in \mathbf{coRP}$. In the same manner, we can define the one-sided error algorithm A_2 and show that $L \in \mathbf{RP}$. Therefore $L \in \mathbf{RP} \cap \mathbf{coRP}$. $\square$

Lemma: $\mathbf{RP} \cap \mathbf{coRP} \subseteq \mathbf{ZPP}$.

Proof: Let $L \in \mathbf{RP} \cap \mathbf{coRP}$. It means

- There is a randomized algorithm A_1 with polynomial time complexity where

  $x \in L \Rightarrow Pr[A_1(x) \text{ accepts}] \geq \frac{1}{2}$ and
  $x \notin L \Rightarrow Pr[A_1(x) \text{ accepts}] = 0$

- There is a randomized algorithm A_2 with polynomial time complexity where

  $x \in L \Rightarrow Pr[A_2(x) \text{ accepts}] = 1$ and
  $x \notin L \Rightarrow Pr[A_2(x) \text{ accepts}] \leq \frac{1}{2}$

We runs both algorithms A_1 and A_2 on input x. We act according to the following table.

E1 A1 accepts x A2 accepts x stop $\boxed{x \in L}$	E2 A1 accepts x A2 rejects x stop $\boxed{x \in L}$
E3 A1 rejects x A2 accepts x Repeat	E4 A1 rejects x A2 rejects x stop $\boxed{x \notin L}$

$$x \in L \Rightarrow Pr[A_1(x) \text{ accepts}] \geq \tfrac{1}{2}$$
$$x \notin L \Rightarrow Pr[A_1(x) \text{ accepts}] = 0$$

$$x \in L \Rightarrow Pr[A_2(x) \text{ accepts}] = 1$$
$$x \notin L \Rightarrow Pr[A_2(x) \text{ accepts}] \leq \tfrac{1}{2}$$

E_3 (repeat) happens with probability at most $\frac{1}{2}$.

$$Pr[A_1 \text{ rejects and } A_2 \text{ accepts} \,|x \in L] \leq \frac{1}{2} \times 1 = \frac{1}{2}$$

$$Pr[A_1 \text{ rejects and } A_2 \text{ accepts} \,|x \notin L] \leq 1 \times \frac{1}{2} = \frac{1}{2}$$

$$Pr[E_3] = Pr[A_1 \text{ rejects and } A_2 \text{ accepts}] \leq \frac{1}{2}$$

Recall: Good event happens with probability p. Bad event happens with probability $1-p$. Expected number of repetitions until a good event happens is most $\frac{1}{p}$.

Expected number of repetition until $E_1 \cup E_2 \cup E_4$ happens for the first time is most $\frac{1}{1/2} = 2$

Let $f(n) = \max\{\text{running time of } A_1, \text{ running time of } A_2\}$

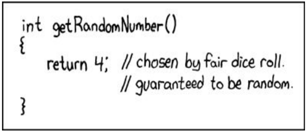
Let R be the running time of the proposed algorithm on input x. We have

$E[R] \leq 2f(n) \times \text{ expected number of repetitions } \leq 2f(n) \times 2$

Therefore the proposed algorithm has $O(f(n))$ expected running time and always gives the correct answer.

Therefore $L \in \mathbf{ZPP}$.

a random joke



Randomized approximation algorithm for 3-SAT

Input: A formula ϕ with $3 - CNF$ format consisting of m clauses defined over n variables.

$$(x_1 \vee \overline{x_4} \vee x_3) \wedge (x_2 \vee \overline{x_4} \vee \overline{x_1}) \wedge \ldots \wedge$$

Problem: Find an assignment that satisfy the most number of clauses.

Algorithm I: Set each variable $x_1, \ldots, x_n$ independently to <u>False</u> or <u>True</u> with probability $\frac{1}{2}$ each.

What is the expected number of clauses satisfied by such random assignment?

Let $Z_i = 1$ if clause C_i is satisfied otherwise $Z_i = 0$.

$\text{number of satisfied clauses} = \sum_{i=1}^{m} Z_i =$

$\text{expected number of satisfied clauses} = E[\sum_{i=1}^{m} Z_i] = \sum_{i=1}^{m} E[Z_i] = \sum_{i=1}^{m} Pr[Z_i = 1] = \frac{7}{8}m$

- **Observation**: Given any 3-CNF formula ϕ there is always an assignment that satisfy at least $\frac{7}{8}$ number of clauses in ϕ
- Follows from the fact that when $E[X] = k$ there is an event where $X \geq k$.
- We want to find an assignment that satisfy $\frac{7}{8}$ of the clauses.
- Let p be the probability that a random assignment satisfy $\frac{7}{8}$ of the clauses?
- We prove a lower bound on p

- Let p_j denote the probability that a random assignment satisfy exactly j clauses
- expected number of satisfied clauses $= \sum_{j=1}^{m} jp_j$
- $\sum_{j=1}^{m} jp_j = \frac{7}{8}m$
- $p = \sum_{j \geq \frac{7}{8}m} p_j \qquad 1 - p = \sum_{j < \frac{7}{8}m} p_j$
- $\frac{7}{8}m = \sum_{j < \frac{7}{8}m} jp_j + \sum_{j \geq \frac{7}{8}m} jp_j$
- Let m' be the largest number smaller than $\frac{7}{8}m$
- $\frac{7}{8}m \leq \sum_{j < \frac{7}{8}m} m'p_j + \sum_{j \geq \frac{7}{8}m} mp_j = m' \sum_{j < \frac{7}{8}m} p_j + m \sum_{j \geq \frac{7}{8}m} p_j$
- $\frac{7}{8}m \leq m'(1-p) + mp$

- $p \geq \frac{\frac{7}{8}m - m'}{m} \geq \frac{\frac{1}{8}}{m} \geq \frac{1}{8m}$

- p is the probability that a random assignment satisfy at least $\frac{7}{8}$ of the clauses.

> **Algorithm II**: Set each variable $x_1, \ldots, x_n$ independently to <u>False</u> or <u>True</u> with probability $\frac{1}{2}$ each. Check if $\frac{7}{8}$ of the clauses are satisfied. If not, repeat.

- expected number of trails before success $\leq \frac{1}{p} \leq 8m$

- Algorithm II has expected running time $O(m(n+m))$ and, given formula ϕ, finds an assignment that satisfy at least $\frac{7}{8}$ of the clauses in ϕ

Two-sided error algorithms: class PP

Definition: The class **PP** (Probabilistic Polynomial time) consists of all languages L that have a randomized algorithm A that runs in polynomial time such that for any input $x \in \Sigma^*$

- $x \in L \Rightarrow Pr[A(x) \text{ accepts}] > \frac{1}{2}$
- $x \notin L \Rightarrow Pr[A(x) \text{ accepts}] < \frac{1}{2}$

- To boost the probability of success (for example to $\frac{1}{4}$), we can repeat the algorithm and output <u>the majority</u> of the answers.

- Number of required repetitions could be exponential!

- Not very practical.

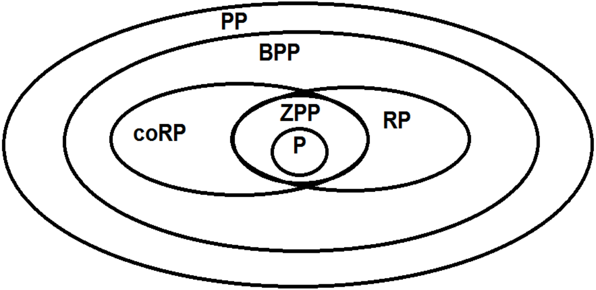
Two-sided error algorithms: class **BPP**

Definition: The class **BPP** (Bounded-error Probabilistic Polynomial time) consists of all languages L that have a randomized algorithm A that runs in polynomial time such that for any input $x \in \Sigma^*$

- $x \in L \Rightarrow Pr[A(x) \text{ accepts}] > \frac{3}{4}$
- $x \notin L \Rightarrow Pr[A(x) \text{ accepts}] < \frac{1}{4}$

Given $x \in \Sigma^*$, the probability of error of the algorithm on input x, is at most $\frac{1}{4}$.

Fact: The error probability of the algorithm can be reduced to $\frac{1}{2^n}$ by polynomial number of repetitions.



Conjecture: $\mathrm{BPP} = \mathrm{P}$