Лекция 2

РАНДОМИЗАЦИЯ И ТЕОРИЯ ОТПЕЧАТКОВ

1. Отпечатки для идентификации

2. Как работают антивирусные программы

В последнее время компьютерные вирусы стали настоящим "*бедствием*" для всех, кто пользуется компьютерами. С целью обнаружения и профилактики "*вирусов*" используются *антивирусные программы.* Одним из методов, которым антивирусные программы обнаруживают вредоносное программное обеспечение, является сканирование файлов для поиска известных вирусов. Анализируя файлы, антивирусная программа обращается к антивирусным базам, составленным производителем программы-антивируса. В случае соответствия какого-либо участка кода просматриваемого файла сигнатуре вируса в базах программа-антивирус может по запросу выполнить одно из действий, включая удаление или лечение файла. Поскольку в базах данных содержатся тысячи известных вирусов, алгоритмы поиска совпадений эталонов и участков кода является одной из актуальных задач.

2.1. Поиск шаблона в файле. Задача обнаружения вирусов, внедрившихся в тело файла, называется в компьютерных науках *поиском шаблона в файле.*

Рассмотрим классическую задачу поиска шаблона Y в файле X. Мы хотим определить встречается ли шаблон $Y = y_1 y_2 \dots y_m$ в файле $X = x_1 x_2 \dots x_n$, $n > m$.

Непосредственный способ последовательных сравнений шаблона с участками файла выполняется за $O((n-m)m)$ операций. Существует сложный детерминированный алгоритм, который выполняется за время $O(n+m)$ (понятно, что это оптимальный результат). Существует простой рандомизированный алгоритм Карпа–Рабина, который также выполняется за время $O(n+m)$ и основан на той же идее, что и алгоритм, представленный в разделе 2.

Для простоты считаем, что алфавит бинарный, то есть X и Y состоят из нулей и единиц (из *битов*). Пусть $X(j) = x_j x_{j+1} \ldots x_{j+m-1}$ обозначает участок файла X длины m, начиная с позиции j. Отпечаток участка $X(j)$ мы определяем по формуле

$$\textsc{fing}(X(j)) = g(x_j \ldots x_{j+m-1}) \bmod p,$$

где

$$\begin{aligned} g(x_j \ldots x_{j+m-1}) & \\ &= x_j 2^{m-1} + x_{j+1} 2^{m-2} + \cdots + x_{j+m-2} 2^1 + x_{j+m-1} 2^0 \\ &= \sum_{k=j}^{j+m-1} x_k 2^{m-1+j-k}. \end{aligned}$$

Аналогично определяется и $\textsc{fing}(Y)$.

Ниже приведен алгоритм Карпа–Рабина.

Input: Файл X, $|X| = n$; шаблон Y, $|Y| = m$;
достаточно большое число c.
Output: Индикатор `найден/не найден` шаблон в файле
1. Вычислить $k = cmn \ln(cmn)$;
2. Выбрать случайное простое число p из $\{1, \ldots, k\}$;
3. `for` $j = 1$ `to` $n - m + 1$ `do`
`if` $\textsc{fing}(X(j)) = \textsc{fing}(Y)$ `then` шаблон найден `stop`

Алгоритм 1. Рандомизированный поиск шаблона

Если алгоритм заканчивается, а оператор **stop** не срабатывает, то записи Y действительно нет в файле X. Ошибка может возникнуть, если алгоритм определит наличие шаблона (ведь два разных числа могут иметь одинаковые остатки от деления на p).

Чтобы определить вероятность ошибки, мы рассуждаем так же, как и в разделе 2. Пусть j фиксировано и $X(j) \neq Y$. Тогда абсолютная величина числа $X(j) - Y$ не превосходит 2^m, значит оно имеет не больше, чем m простых делителей. Кроме того, $\textsc{fing}(X(j)) = \textsc{fing}(Y)$ тогда и только тогда, когда p делит $X(j) - Y$. Поэтому

$$\mathsf{P}(\textsc{fing}(X(j)) = \textsc{fing}(Y)) \leq \frac{m}{\pi(k)}. \tag{1}$$

Если же $X(j) \neq Y$ для всех $1 \leq j \leq n - m + 1$, то

$$\begin{aligned} &\mathsf{P}(\text{алгоритм “находит” шаблон}) \\ &\quad = \mathsf{P}\left(\bigcup_{j=1}^{n-m+1} \{\textsc{fing}(X(j)) = \textsc{fing}(Y)\}\right) \\ &\quad \leq \frac{(n-m+1)m}{\pi(k)} \sim \frac{1}{c}, \end{aligned} \tag{2}$$

если выбрать $k = cnm \ln(cnm)$ (рассуждения при доказательстве эквивалентности $\sim$ такие же, как и в разделе 2).

Время выполнения алгоритма можно оценить “*в лоб*”: каждая итерация цикла требует вычисления отпечатка числа, состоящего из m бит, а это занимает время $O(m)$, что означает, что общее время выполнения равно $O(nm)$.

Однако, на самом деле, эту оценку можно заметно улучшить, заметив, что

$$
\begin{aligned}
g(X(j+1)) &= \sum_{k=j+1}^{j+m} x_k 2^{m+j-k} \\
&= x_{j+m} + \sum_{k=j}^{j+m-1} x_k 2^{m+j-k} - x_j 2^m \qquad (3)\\
&= 2(g(X(j)) - 2^{m-1}x_j) + x_{j+m}.
\end{aligned}
$$

Каждая итерация в (3) выполняется за время $O(1)$ (фактически необходимы 2 сложения и 2 умножения).

Чтобы запустить итерационный процесс, необходимо вычислить $\textsc{fing}(X(1))$ и $\textsc{fing}(Y)$. Для этого требуется $O(m)$ операций. Поэтому весь алгоритм выполняется за время $O(n+m)$, как и утверждалось выше.

3. Проверка равенства матриц

Вычисление произведения матриц играет ключевую роль во многих вычислениях. Размеры перемножаемых матриц могут быть чрезвычайно велики, поэтому эффективность операции умножения играет ключевую роль в подобных вычислениях. Такую же важную роль играет проверка вычислений, которая сводится к проверке равенства матриц $AB = C$.

Пусть заданы три матрицы A, B, C размера $n \times n$, элементами которых являются только 0 или 1. Мы рассматриваем арифметические операции по модулю 2, то есть

$$
x + y = \begin{cases} 0, & \text{если } x = y = 0 \text{ или } x = y = 1, \\ 1, & \text{если } x = 0,\ y = 1 \text{ или } x = 0,\ y = 1. \end{cases}
$$

Как наиболее эффективно проверить матричное равенство

$$AB = C?$$

Простой и одновременно естественный способ состоит из двух шагов:

1. перемножить матрицы A и B;
2. проверить равенство для каждого из элементов матриц AB и C.

Для вычисления произведения матриц A и B требуется $O(n^3)$ операций: элементы матрицы AB вычисляются по формуле

$$(AB)_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \bmod 2, \qquad 1 \le i, j \le n.$$

Таким образом, для вычисления элемента $(AB)_{ij}$ требуется n операций умножения и n операций сложения. Поскольку матрица размера $n \times n$ имеет n^2 элементов, то для вычисления произведения AB действительно требуется $O(n^3)$ операций. Существует остроумный детерминированный метод, который позволяет осуществить операцию умножения двух матриц за $O(n^{2.375})$ операций.

Мы рассмотрим рандомизированный алгоритм сравнения матриц AB и C, который нуждается в $O(n^2)$ операциях. Этот алгоритм позволяет утверждать, что равенство $AB = C$ выполнено лишь с определенной вероятностью, которую можно оценить довольно точно.

3.1. Рандомизированный алгоритм проверки равенства матриц. Сначала алгоритм генерирует случайный вектор $\bar{\mathbf{r}}$ размерности n, который имеет равномерное распределение в $\{0, 1\}^n$. Это означает, что координаты вектора $\bar{\mathbf{r}}$ равны либо 0, либо 1, причем

$$\mathsf{P}(\bar{\mathbf{r}} = \bar{\varepsilon}) = \frac{1}{2^n}$$

для любого вектора $\bar{\varepsilon}$, составленного из 0 и 1.

Затем алгоритм вычисляет последовательно векторы

$$B\overline{\mathbf{r}}, \qquad A(B\overline{\mathbf{r}}), \qquad C\overline{\mathbf{r}}.$$

Работа алгоритма заканчивается проверкой равенства

$$A(B\overline{\mathbf{r}}) = C\overline{\mathbf{r}}.$$

Если это равенство нарушается, то принимается верное решение о $AB \neq C$. В противном же случае алгоритм решает, что $AB = C$, а это уже верно не всегда. Поскольку $\overline{\mathbf{r}}$ — случайный вектор, то ошибочное решение также случайно; его вероятность можно оценить.

Ниже приведена блок-схема данного алгоритма.

Input: Матрицы A, B, C размера $n \times n$.
Output: $d = 1$, если $AB = C$; или $d = 0$, если $AB \neq C$.
1. Сгенерировать случайный вектор $\overline{\mathbf{r}} = (r_1, \ldots, r_n) \in \{0,1\}^n$.
2. Вычислить $B\overline{\mathbf{r}}$, $A(B\overline{\mathbf{r}})$, $C\overline{\mathbf{r}}$.
3. Сравнить $A(B\overline{\mathbf{r}})$ и $C\overline{\mathbf{r}}$:
    `if` $A(B\overline{\mathbf{r}}) = C\overline{\mathbf{r}}$ `then` $d = 1$ `else` $d = 0$.
4. `stop`.

Алгоритм 2. Сравнение матриц

3.2. Время генерирования вектора. Для генерации вектора $\overline{\mathbf{r}}$, имеющего равномерное распределение в $\{0,1\}^n$, необходимо $O(n)$ операций. Это вытекает из следующего результата.

Лемма 1. *Вектор $\overline{\mathbf{r}}$ имеет равномерное распределение в множестве $\{0,1\}^n$ тогда и только тогда, когда его координаты r_k, $1 \leq k \leq n$, независимы и имеют симметричное распределение Бернулли:*

$$\mathsf{P}(r_k = 0) = \mathsf{P}(r_k = 1) = \frac{1}{2}.$$

Из леммы вытекает, что генерирование вектора $\overline{\mathbf{r}}$, имеющего равномерное распределение в $\{0,1\}^n$, сводится к генерированию n случайных величин Бернулли. Поскольку генерирование одной случайной величины Бернулли требует $O(1)$ операций, то генерирование вектора $\overline{\mathbf{r}}$ действительно требует $O(n)$ операций.

3.3. Время вычисления произведения матрицы на вектор. Элементы векторов $B\overline{\mathbf{r}}$, $A(B\overline{\mathbf{r}})$ и $C\overline{\mathbf{r}}$ вычисляются по обычной формуле, которую запишем для $B\overline{\mathbf{r}}$:

$$(B\overline{\mathbf{r}})_i = \sum_{k=1}^{n} b_{ik} r_k \bmod 2, \qquad 1 \le i \le n.$$

Вычисление произведение матрицы размера $n \times n$ на вектор размерности n требует $O(n^2)$ операций, поскольку вычисление каждого элемента произведения требует n операций умножения и n операций сложения. Поэтому $O(n^2)$ операций достаточно для вычисления всех векторов $B\overline{\mathbf{r}}$, $A(B\overline{\mathbf{r}})$, $C\overline{\mathbf{r}}$.

3.4. Общее время. Сравнение двух векторов требует $O(n)$ операций. Следовательно, алгоритм заканчивает работу за $O(n) + O(n^2) + O(n) = O(n^2)$ операций.

3.5. Вероятность ошибки алгоритма. Если на самом деле $AB = C$, то $A(B\overline{\mathbf{r}}) = C\overline{\mathbf{r}}$ для любого $\overline{\mathbf{r}}$. Если же $AB \ne C$, то тем не менее равенство $A(B\overline{\mathbf{r}}) = C\overline{\mathbf{r}}$ все же может выполняться для некоторых векторов $\overline{\mathbf{r}}$. Если такое случается, то алгоритм дает неверный ответ. Вероятность неверного ответа оценивается в следующей теореме.

Теорема 1. *Если $AB \ne C$, а вектор $\overline{\mathbf{r}}$ имеет равномерное распределение в множестве $\{0,1\}^n$, то*

$$\mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}) \le \frac{1}{2}.$$

3.6. Как сделать вероятность ошибки малой. Улучшить оценку вероятности ошибки, полученную в теореме 1, невозможно. Поэтому кажется, что пользоваться описанным методом крайне опасно при проверке равенства двух матриц: ошибка может наблюдаться при каждой второй проверке. Однако минимальным усилием ситуация существенно улучшается. Именно, повторим описанную процедуру еще раз. При этом, естественно, случайный вектор $\overline{\mathbf{r}}$ будет другим. Обозначим эти два вектора, возникающие при первом и втором повторе процедуры, через $\overline{\mathbf{r}}'$ и $\overline{\mathbf{r}}''$. Поскольку векторы $\overline{\mathbf{r}}'$ и $\overline{\mathbf{r}}''$ независимы, вероятность двойной ошибки можно оценить так:

$$\begin{aligned}&\mathsf{P}(AB\overline{\mathbf{r}}' = C\overline{\mathbf{r}}', AB\overline{\mathbf{r}}'' = C\overline{\mathbf{r}}'')\\ &\qquad = \mathsf{P}(AB\overline{\mathbf{r}}' = C\overline{\mathbf{r}}')\mathsf{P}(AB\overline{\mathbf{r}}'' = C\overline{\mathbf{r}}'')\\ &\qquad \le \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4}.\end{aligned}$$

Повторяя процедура 10 раз, снижаем вероятность ошибки до $\frac{1}{2^{10}} \le \frac{1}{1000}$. Если необходимо, чтобы вероятность ошибки не превосходила $\delta > 0$, то достаточно повторить процедуру $1 + \left[\log_2 \delta^{-1}\right]$ раз. Такое повторение нисколько не влияет на оценку $O(n^2)$ времени выполнения процедуры.

Доказательства. Сначала докажем лемму 1, а потом теорему 1.

Доказательство леммы 1. $\lhd$ Пусть $r_1, \ldots, r_n$ — независимые случайные величины Бернулли. Докажем, что вектор $\overline{\mathbf{r}} = (r_1, \ldots, r_n)$ имеет равномерное распределение на множестве $\{0,1\}^n$. Выберем произвольный вектор $\overline{\boldsymbol{\varepsilon}} \in \{0,1\}^n$, то есть $\overline{\boldsymbol{\varepsilon}} = (\varepsilon_1, \ldots, \varepsilon_n)$, где $\varepsilon_i \in \{0,1\}$, $1 \le i \le n$. Тогда в силу независимости

$$\begin{aligned}\mathsf{P}(\overline{\mathbf{r}} = \overline{\boldsymbol{\varepsilon}}) &= \mathsf{P}(r_1 = \varepsilon_1, \ldots, r_n = \varepsilon_n)\\ &= \mathsf{P}(r_1 = \varepsilon_1)\ldots\mathsf{P}(r_n = \varepsilon_n) = \frac{1}{2^n}.\end{aligned}$$

Это и означает, что вектор $\bar{\mathbf{r}}$ имеет равномерное распределение на множестве $\{0,1\}^n$.

$\triangleright$ Пусть теперь вектор $\bar{\mathbf{r}}$ имеет равномерное распределение на множестве $\{0,1\}^n$. Докажем, что его координаты $r_1,\dots,r_n$ независимы и имеют распределение Бернулли. С целью демонстрации идеи рассмотрим сначала случай $n=2$: для $\varepsilon_1 \in \{0,1\}$ имеем

$$\begin{aligned}\mathsf{P}(r_1=\varepsilon_1) &= \mathsf{P}(r_1=\varepsilon_1,\varepsilon_2=0)+\mathsf{P}(r_1=\varepsilon_1,\varepsilon_2=1)\\ &= \frac{1}{4}+\frac{1}{4}=\frac{1}{2},\end{aligned}$$

то есть r_1 — случайная величина Бернулли. Аналогично доказывается бернуллиевость величины r_2. Теперь их независимость доказать просто:

$$\mathsf{P}(r_1=\varepsilon_1,r_2=\varepsilon_2)=\frac{1}{4}=\frac{1}{2}\cdot\frac{1}{2}=\mathsf{P}(r_1=\varepsilon_1)\cdot\mathsf{P}(r_2=\varepsilon_2).$$

Доказательство в общем случае совершенно аналогично: если $\varepsilon_1 \in \{0,1\}$, то

$$\begin{aligned}\mathsf{P}(r_1=\varepsilon_1) &= \sum_{\varepsilon_2,\dots,\varepsilon_n\in\{0,1\}} \mathsf{P}(r_1=\varepsilon_1,\dots,r_n=\varepsilon_n)\\ &= \sum_{\varepsilon_2,\dots,\varepsilon_n\in\{0,1\}} \frac{1}{2^n}=2^{n-1}\cdot\frac{1}{2^n}=\frac{1}{2}.\end{aligned}$$

Это и означает, что r_1 — случайная величина Бернулли. Аналогично доказывается бернуллиевость других величины $r_2,\dots,r_n$. Для доказательства независимости $r_1,\dots,r_k$ выберем $\varepsilon_1,\dots,\varepsilon_k \in \{0,1\}$ и запишем

$$\begin{aligned}&\mathsf{P}(r_1=\varepsilon_1,\dots,r_k=\varepsilon_k)\\ &= \sum_{\varepsilon_{k+1},\dots,\varepsilon_n\in\{0,1\}} \mathsf{P}(r_1=\varepsilon_1,\dots,r_k=\varepsilon_k,r_{k+1}=\varepsilon_{k+1},r_n=\varepsilon_n)\\ &= 2^{n-k}\cdot\frac{1}{2^n}=\frac{1}{2^k}=\mathsf{P}(r_1=\varepsilon_1)\dots\mathsf{P}(r_k=\varepsilon_k).\end{aligned}$$

Это и означает, что случайные величины $r_1, \dots, r_k$ независимы. Доказательство независимости других комбинаций из набора $r_1, \dots, r_n$ абсолютно аналогично. $\square$

Доказательство теоремы 1. Положим $D \stackrel{\text{def}}{=} AB - C$. Если $AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}$, то $D\overline{\mathbf{r}} = 0$. Если при этом $D \neq 0$, то у матрицы D имеется ненулевой элемент. Не теряя общности считаем, что $d_{11} \neq 0$. Из $D\overline{\mathbf{r}} = 0$ вытекает

$$\sum_{j=1}^{n} d_{1j} r_j = 0,$$

откуда

$$r_1 = -\frac{1}{d_{11}} \sum_{j=2}^{n} d_{1j} r_j. \tag{4}$$

Чтобы лучше представить дальнейшую оценку вероятности $\mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}})$, рассмотрим сначала случай $n = 2$. Тогда (4) равносильно

$$r_1 = -\frac{d_{12}}{d_{11}} r_2 = \theta r_2, \qquad \text{где} \qquad \theta \stackrel{\text{def}}{=} -\frac{d_{12}}{d_{11}}. \tag{5}$$

Поскольку (5) вытекает из равенства $AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}$, то

$$\{\omega \colon AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}\} \subseteq \{\omega \colon r_1 = \theta r_2\}.$$

Следовательно,

$$\begin{aligned}
\mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}) &= \mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}, r_2 = 0) + \mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}, r_2 = 1) \\
&\le \mathsf{P}(r_1 = \theta r_2, r_2 = 0) + \mathsf{P}(r_1 = \theta r_2, r_2 = 1) \\
&= \mathsf{P}(r_1 = 0, r_2 = 0) + \mathsf{P}(r_1 = \theta, r_2 = 1).
\end{aligned}$$

В силу независимости величин r_1 и r_2 имеем

$$\mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}) \le \mathsf{P}(r_1 = 0) \cdot \mathsf{P}(r_2 = 0) + \mathsf{P}(r_1 = \theta) \cdot \mathsf{P}(r_2 = 1).$$

Ясно, что $\mathsf{P}(r_1 = \theta) \le \frac{1}{2}$, причем равенство возможно только при $\theta = 0$ или $\theta = 1$ (в остальных случаях эта вероятность равна 0). Поэтому первый сомножитель у слагаемых в правой части последнего неравенство не превосходит $\frac{1}{2}$, откуда и вытекает

$$\mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}) \le \frac{1}{2}(\mathsf{P}(r_2 = 0) + \mathsf{P}(r_2 = 1)) = \frac{1}{2}.$$

Доказательство для других n абсолютно аналогично: положим

$$\xi \stackrel{\text{def}}{=} -\frac{1}{d_{11}} \sum_{j=2}^{n} d_{1j} r_j.$$

Тогда $\{AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}\} \subseteq \{r_1 = \xi\}$ и из (4) получаем

$$\begin{aligned} &\mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}) \\ &= \sum_{\varepsilon_2,\ldots,\varepsilon_n \in \{0,1\}} \mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}, r_2 = \varepsilon_2, \ldots, r_n = \varepsilon_n) \\ &\le \sum_{\varepsilon_2,\ldots,\varepsilon_n \in \{0,1\}} \mathsf{P}(r_1 = \xi, r_2 = \varepsilon_2, \ldots, r_n = \varepsilon_n). \end{aligned}$$

На событии $\{r_1 = \xi, r_2 = \varepsilon_2, \ldots, r_n = \varepsilon_n\}$ величина ξ равна

$$x \stackrel{\text{def}}{=} -\frac{1}{d_{11}} \sum_{j=2}^{n} d_{1j} \varepsilon_j$$

(x отличается от ξ тем, что каждая величина r_j, $2 \le j \le n$, в определении ξ заменяется на число ε_j). В силу независимости случайной величини r_1 и случайного вектора $(r_2, \ldots, r_n)$

оценки можно продолжить следующим образом:

$$\begin{aligned}
&\mathsf{P}(AB\overline{\mathbf{r}} = C\overline{\mathbf{r}}) \\
&\qquad \le \sum_{\varepsilon_2,\ldots,\varepsilon_n \in \{0,1\}} \mathsf{P}(r_1 = x, r_2 = \varepsilon_2, \ldots, r_n = \varepsilon_n) \\
&\qquad = \sum_{\varepsilon_2,\ldots,\varepsilon_n \in \{0,1\}} \mathsf{P}(r_1 = x)\mathsf{P}(r_2 = \varepsilon_2, \ldots, r_n = \varepsilon_n) \\
&\qquad \le \frac{1}{2} \sum_{\varepsilon_2,\ldots,\varepsilon_n \in \{0,1\}} \mathsf{P}(r_2 = \varepsilon_2, \ldots, r_n = \varepsilon_n) \\
&\qquad = \frac{1}{2}.
\end{aligned}$$

$\square$