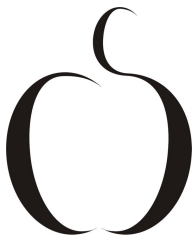


с/к “Эффективные алгоритмы”

Лекция 9: Вероятностные алгоритмы

А. Куликов

Computer Science клуб при ПОМИ
<http://logic.pdmi.ras.ru/~infclub/>



План лекции

- 1 Проверка перемножения матриц

План лекции

- 1 Проверка перемножения матриц
- 2 Сравнение строк

План лекции

- 1 Проверка перемножения матриц
- 2 Сравнение строк
- 3 Нулевой многочлен

План лекции

- 1 Проверка перемножения матриц
- 2 Сравнение строк
- 3 Нулевой многочлен
- 4 Эквивалентность ветвящихся программ

План лекции

- 1 Проверка перемножения матриц
- 2 Сравнение строк
- 3 Нулевой многочлен
- 4 Эквивалентность ветвящихся программ

Проверка перемножения матриц

Проверка перемножения матриц

Проверка перемножения матриц

Проверка перемножения матриц

- Даны три $n \times n$ матрицы A , B , C . Мы хотим проверить равенство $A \times B = C$.

Проверка перемножения матриц

Проверка перемножения матриц

- Даны три $n \times n$ матрицы A , B , C . Мы хотим проверить равенство $A \times B = C$.
- Простой детерминированный алгоритм перемножает матрицы A и B и сравнивает результат с C .

Проверка перемножения матриц

Проверка перемножения матриц

- Даны три $n \times n$ матрицы A , B , C . Мы хотим проверить равенство $A \times B = C$.
- Простой детерминированный алгоритм перемножает матрицы A и B и сравнивает результат с C .
- Время работы такого алгоритма при использовании обычного перемножения матриц составляет $O(n^3)$, при использовании быстрого — $O(n^{2.376})$.

Проверка перемножения матриц

Проверка перемножения матриц

- Даны три $n \times n$ матрицы A , B , C . Мы хотим проверить равенство $A \times B = C$.
- Простой детерминированный алгоритм перемножает матрицы A и B и сравнивает результат с C .
- Время работы такого алгоритма при использовании обычного перемножения матриц составляет $O(n^3)$, при использовании быстрого — $O(n^{2.376})$.
- Мы предъявим вероятностный алгоритм с **односторонней ошибкой**, который за время $O(n^2)$ проверяет равенство.

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A , B , C)

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A , B , C)

- взять случайный вектор x из $\{0, 1\}^n$

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A, B, C)

- взять случайный вектор x из $\{0, 1\}^n$
- посчитать $y = B \times x$

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A , B , C)

- взять случайный вектор x из $\{0, 1\}^n$
- посчитать $y = B \times x$
- посчитать $z = A \times y$

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A , B , C)

- взять случайный вектор x из $\{0, 1\}^n$
- посчитать $y = B \times x$
- посчитать $z = A \times y$
- посчитать $t = C \times x$

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A , B , C)

- взять случайный вектор x из $\{0, 1\}^n$
- посчитать $y = B \times x$
- посчитать $z = A \times y$
- посчитать $t = C \times x$
- вернуть “да”, если $z = t$

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A , B , C)

- взять случайный вектор x из $\{0, 1\}^n$
- посчитать $y = B \times x$
- посчитать $z = A \times y$
- посчитать $t = C \times x$
- вернуть “да”, если $z = t$
- вернуть “нет”

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A , B , C)

- взять случайный вектор x из $\{0, 1\}^n$
- посчитать $y = B \times x$
- посчитать $z = A \times y$
- посчитать $t = C \times x$
- вернуть “да”, если $z = t$
- вернуть “нет”

Неформально

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A , B , C)

- взять случайный вектор x из $\{0, 1\}^n$
- посчитать $y = B \times x$
- посчитать $z = A \times y$
- посчитать $t = C \times x$
- вернуть “да”, если $z = t$
- вернуть “нет”

Неформально

- Алгоритм просто проверяет равенство $A \times (B \times x) = C \times x$.

Алгоритм

Алгоритм

RAND-MATRIX-EQUIV(A, B, C)

- взять случайный вектор x из $\{0, 1\}^n$
- посчитать $y = B \times x$
- посчитать $z = A \times y$
- посчитать $t = C \times x$
- вернуть “да”, если $z = t$
- вернуть “нет”

Неформально

- Алгоритм просто проверяет равенство $A \times (B \times x) = C \times x$.
- Матрицы могут быть над любым полем.

Анализ алгоритма

Лемма

Алгоритм RAND-MATRIX-EQUIV имеет время работы $O(n^2)$ и вероятность ошибки не более $1/2$. Причем ошибиться он может, только если $A \times B \neq C$.

Доказательство

Анализ алгоритма

Лемма

Алгоритм RAND-MATRIX-EQUIV имеет время работы $O(n^2)$ и вероятность ошибки не более $1/2$. Причем ошибиться он может, только если $A \times B \neq C$.

Доказательство

- Предположим, что $A \times B \neq C$.

Анализ алгоритма

Лемма

Алгоритм RAND-MATRIX-EQUIV имеет время работы $O(n^2)$ и вероятность ошибки не более $1/2$. Причем ошибиться он может, только если $A \times B \neq C$.

Доказательство

- Предположим, что $A \times B \neq C$.
- Алгоритм ошибается, если для случайного вектора, который он выбрал, $(A \times B - C) \times x = 0$.

Анализ алгоритма

Лемма

Алгоритм RAND-MATRIX-EQUIV имеет время работы $O(n^2)$ и вероятность ошибки не более $1/2$. Причем ошибиться он может, только если $A \times B \neq C$.

Доказательство

- Предположим, что $A \times B \neq C$.
- Алгоритм ошибается, если для случайного вектора, который он выбрал, $(A \times B - C) \times x = 0$.
- Рассмотрим ненулевую строчку $d = (d_1, \dots, d_n)$ матрицы $A \times B - C$ (такая есть, поскольку $A \times B \neq C$).

Анализ алгоритма

Лемма

Алгоритм `RAND-MATRIX-EQUIV` имеет время работы $O(n^2)$ и вероятность ошибки не более $1/2$. Причем ошибиться он может, только если $A \times B \neq C$.

Доказательство

- Предположим, что $A \times B \neq C$.
- Алгоритм ошибается, если для случайного вектора, который он выбрал, $(A \times B - C) \times x = 0$.
- Рассмотрим ненулевую строчку $d = (d_1, \dots, d_n)$ матрицы $A \times B - C$ (такая есть, поскольку $A \times B \neq C$).
- НУО, $d_1 \neq 0$.

Доказательство (продолжение)

Доказательство

Доказательство (продолжение)

Доказательство

- Произведение этой строчки на x равно $\sum_{i=1}^n d_i x_i$.

Доказательство (продолжение)

Доказательство

- Произведение этой строчки на x равно $\sum_{i=1}^n d_i x_i$.
- Это произведение равно 0 тогда и только тогда, когда $x_1 = h(x_2, \dots, x_n)$, где

$$h(x_2, \dots, x_n) = -\frac{\sum_{i=2}^n d_i x_i}{d_1}.$$

Доказательство (продолжение)

Доказательство

- Произведение этой строчки на x равно $\sum_{i=1}^n d_i x_i$.
- Это произведение равно 0 тогда и только тогда, когда $x_1 = h(x_2, \dots, x_n)$, где

$$h(x_2, \dots, x_n) = -\frac{\sum_{i=2}^n d_i x_i}{d_1}.$$

- Отметим, что h может принимать значения, отличные от 0 и 1 .

Доказательство (продолжение)

Доказательство

- Произведение этой строчки на x равно $\sum_{i=1}^n d_i x_i$.
- Это произведение равно 0 тогда и только тогда, когда $x_1 = h(x_2, \dots, x_n)$, где

$$h(x_2, \dots, x_n) = -\frac{\sum_{i=2}^n d_i x_i}{d_1}.$$

- Отметим, что h может принимать значения, отличные от 0 и 1 .
- С вероятностью хотя бы $1/2$ $x_1 \neq h(x_2, \dots, x_n)$.



План лекции

- 1 Проверка перемножения матриц
- 2 Сравнение строк**
- 3 Нулевой многочлен
- 4 Эквивалентность ветвящихся программ

Сравнение строк

Сравнение строк

Сравнение строк

Сравнение строк

- Даны две битовые строки a и b , которые необходимо сравнить на совпадение, затратив как можно меньше информации (к примеру, требуется сравнить файлы по сети).

Сравнение строк

Сравнение строк

- Даны две битовые строки a и b , которые необходимо сравнить на совпадение, затратив как можно меньше информации (к примеру, требуется сравнить файлы по сети).
- Основная идея: сравнивать не сами строки, а функции от них.

Сравнение строк

Сравнение строк

- Даны две битовые строки a и b , которые необходимо сравнить на совпадение, затратив как можно меньше информации (к примеру, требуется сравнить файлы по сети).
- Основная идея: сравнивать не сами строки, а функции от них.
- Будем сравнивать $a \bmod p$ и $b \bmod p$.

Сравнение строк

Сравнение строк

- Даны две битовые строки a и b , которые необходимо сравнить на совпадение, затратив как можно меньше информации (к примеру, требуется сравнить файлы по сети).
- Основная идея: сравнивать не сами строки, а функции от них.
- Будем сравнивать $a \bmod p$ и $b \bmod p$.
- Для такого сравнения достаточно передать $2 \log_2 p$ битов.

Асимптотический закон распределения простых чисел

Определение

Функция распределения простых чисел (prime distribution function)

$\pi(n)$ определяется как количество простых чисел, не превосходящих n .

Асимптотический закон распределения простых чисел

Определение

Функция распределения простых чисел (prime distribution function) $\pi(n)$ определяется как количество простых чисел, не превосходящих n .

Факт

Асимптотический закон распределения простых чисел:

$$\lim_{n \rightarrow \infty} \frac{\pi(n)}{n / \ln n} = 1.$$

Алгоритм

Алгоритм

RAND-STRING-EQUIV(a , b)

Алгоритм

Алгоритм

RAND-STRING-EQUIV(a, b)

- Пусть $|a| = |b| = n$, $N = n^2 \log_2 n^2$.

Алгоритм

Алгоритм

RAND-STRING-EQUIV(a, b)

- Пусть $|a| = |b| = n$, $N = n^2 \log_2 n^2$.
- Выберем случайно простое число p из интервала $[2..N]$.

Алгоритм

Алгоритм

RAND-STRING-EQUIV(a , b)

- Пусть $|a| = |b| = n$, $N = n^2 \log_2 n^2$.
- Выберем случайно простое число p из интервала $[2..N]$.
- Выдать “да”, если $a \bmod p = b \bmod p$.

Алгоритм

Алгоритм

RAND-STRING-EQUIV(a, b)

- Пусть $|a| = |b| = n$, $N = n^2 \log_2 n^2$.
- Выберем случайно простое число p из интервала $[2..N]$.
- Выдать “да”, если $a \bmod p = b \bmod p$.
- Выдать “нет”.

Алгоритм

Алгоритм

RAND-STRING-EQUIV(a, b)

- Пусть $|a| = |b| = n$, $N = n^2 \log_2 n^2$.
- Выберем случайно простое число p из интервала $[2..N]$.
- Выдать “да”, если $a \bmod p = b \bmod p$.
- Выдать “нет”.

Лемма

Алгоритм RAND-STRING-EQUIV является вероятностным алгоритмом с односторонней ошибкой с вероятностью ошибки $O(1/n)$ и требует передачи $O(\log n)$ битов.

Доказательство

Доказательство

Доказательство

Доказательство

- Количество передаваемых битов:

$$2 \log_2 p \leq 2 \log_2 N = 2 \log_2(n^2 \log_2 n^2) = O(\log_2 n).$$

Доказательство

Доказательство

- Количество передаваемых битов:
$$2 \log_2 p \leq 2 \log_2 N = 2 \log_2(n^2 \log_2 n^2) = O(\log_2 n).$$
- Алгоритм ошибается, только если $a \neq b$, но для выбранного простого числа p выполнено равенство $(a - b) \equiv 0 \pmod p$.

Доказательство

Доказательство

- Количество передаваемых битов:
 $2 \log_2 p \leq 2 \log_2 N = 2 \log_2(n^2 \log_2 n^2) = O(\log_2 n)$.
- Алгоритм ошибается, только если $a \neq b$, но для выбранного простого числа p выполнено равенство $(a - b) \equiv 0 \pmod p$.
- Количество таких простых чисел p есть

$$|\{p \in \mathbb{P} : (a - b) \equiv 0 \pmod p\}| \leq n.$$

Доказательство

Доказательство

- Количество передаваемых битов:
 $2 \log_2 p \leq 2 \log_2 N = 2 \log_2(n^2 \log_2 n^2) = O(\log_2 n)$.
- Алгоритм ошибается, только если $a \neq b$, но для выбранного простого числа p выполнено равенство $(a - b) \equiv 0 \pmod p$.
- Количество таких простых чисел p есть

$$|\{p \in \mathbb{P} : (a - b) \equiv 0 \pmod p\}| \leq n.$$

- Значит, вероятность ошибки есть

$$O\left(\frac{n}{N/\ln N}\right) = O\left(\frac{n(\log n^2 + \log \log n^2)}{n^2 \log n^2}\right) = O\left(\frac{1}{n}\right).$$



План лекции

- 1 Проверка перемножения матриц
- 2 Сравнение строк
- 3 Нулевой многочлен**
- 4 Эквивалентность ветвящихся программ

Нулевой многочлен

Нулевой многочлен

Нулевой многочлен

Нулевой многочлен

- Дан многочлен $f(x_1, \dots, x_n)$, и мы хотим проверить равенство $f = 0$ (f равен 0 при любых значениях переменных).

Нулевой многочлен

Нулевой многочлен

- Дан многочлен $f(x_1, \dots, x_n)$, и мы хотим проверить равенство $f = 0$ (f равен 0 при любых значениях переменных).
- Такая проверка может понадобиться, например, если у нас есть матрица, элементы которой являются многочленами, и нас интересует, равен ли 0 ее определитель. Вычислить определитель явно в такой ситуации трудно.

Нулевой многочлен

Нулевой многочлен

- Дан многочлен $f(x_1, \dots, x_n)$, и мы хотим проверить равенство $f = 0$ (f равен 0 при любых значениях переменных).
- Такая проверка может понадобиться, например, если у нас есть матрица, элементы которой являются многочленами, и нас интересует, равен ли 0 ее определитель. Вычислить определитель явно в такой ситуации трудно.
- Основная идея: подставить случайные значения переменным.

Нулевой многочлен

Нулевой многочлен

- Дан многочлен $f(x_1, \dots, x_n)$, и мы хотим проверить равенство $f = 0$ (f равен 0 при любых значениях переменных).
- Такая проверка может понадобиться, например, если у нас есть матрица, элементы которой являются многочленами, и нас интересует, равен ли 0 ее определитель. Вычислить определитель явно в такой ситуации трудно.
- Основная идея: подставить случайные значения переменным.
- Если полученное значение не равно 0, значит, и исходный многочлен не равен 0.

Нулевой многочлен

Нулевой многочлен

- Дан многочлен $f(x_1, \dots, x_n)$, и мы хотим проверить равенство $f = 0$ (f равен 0 при любых значениях переменных).
- Такая проверка может понадобиться, например, если у нас есть матрица, элементы которой являются многочленами, и нас интересует, равен ли 0 ее определитель. Вычислить определитель явно в такой ситуации трудно.
- Основная идея: подставить случайные значения переменным.
- Если полученное значение не равно 0, значит, и исходный многочлен не равен 0.
- Если же значение равно 0, то мы попали в корень этого многочлена.

Более формально

Более формально

Более формально

Более формально

- Пусть \mathbb{F} — поле, а $f(x_1, \dots, x_n)$ — многочлен от n переменных над этим полем.

Более формально

Более формально

- Пусть \mathbb{F} — поле, а $f(x_1, \dots, x_n)$ — многочлен от n переменных над этим полем.
- **Степенью** f называется максимальная степень переменной в записи f через сумму мономов (произведений переменных).

Более формально

Более формально

- Пусть \mathbb{F} — поле, а $f(x_1, \dots, x_n)$ — многочлен от n переменных над этим полем.
- **Степенью** f называется максимальная степень переменной в записи f через сумму мономов (произведений переменных).
- **Полной степенью** называется максимальная степень монома.

Более формально

Более формально

- Пусть \mathbb{F} — поле, а $f(x_1, \dots, x_n)$ — многочлен от n переменных над этим полем.
- **Степенью** f называется максимальная степень переменной в записи f через сумму мономов (произведений переменных).
- **Полной степенью** называется максимальная степень монома.
- Например, многочлен $f(x_1, x_2, x_3) = x_1^3 x_2 + x_1 x_2^2 x_3^2$ имеет степень 3 и полную степень 5.

Более формально

Более формально

- Пусть \mathbb{F} — поле, а $f(x_1, \dots, x_n)$ — многочлен от n переменных над этим полем.
- **Степенью** f называется максимальная степень переменной в записи f через сумму мономов (произведений переменных).
- **Полной степенью** называется максимальная степень монома.
- Например, многочлен $f(x_1, x_2, x_3) = x_1^3 x_2 + x_1 x_2^2 x_3^2$ имеет степень 3 и полную степень 5.
- Многочлен степени 1 называется **полилинейным**.

Более формально

Более формально

- Пусть \mathbb{F} — поле, а $f(x_1, \dots, x_n)$ — многочлен от n переменных над этим полем.
- **Степенью** f называется максимальная степень переменной в записи f через сумму мономов (произведений переменных).
- **Полной степенью** называется максимальная степень монома.
- Например, многочлен $f(x_1, x_2, x_3) = x_1^3 x_2 + x_1 x_2^2 x_3^2$ имеет степень 3 и полную степень 5.
- Многочлен степени 1 называется **полилинейным**.

Лемма

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен полной степени d над полем \mathbb{F} . Пусть также $S \subseteq \mathbb{F}$ и $|S| \geq d$. Тогда есть хотя бы $|S|^n - d \cdot |S|^{n-1}$ точек $(a_1, \dots, a_n) \in S^n$, в которых $f(a_1, \dots, a_n) \neq 0$.

Доказательство

Доказательство

Доказательство

Доказательство

- Индукция по n .

Доказательство

Доказательство

- Индукция по n .
- База ($n = 1$) верна, поскольку у многочлена степени d не более d корней.

Доказательство

Доказательство

- Индукция по n .
- База ($n = 1$) верна, поскольку у многочлена степени d не более d корней.
- Разложим f по степеням x_n :

$$f = f_0 + f_1 x_n + f_2 x_n^2 + \dots + f_t x_n^t,$$

где f_0, \dots, f_t — многочлены от переменных x_1, \dots, x_{n-1} , $f_t \neq 0$, $t \leq d$.

Доказательство (продолжение)

Доказательство

Доказательство (продолжение)

Доказательство

- Нам нужно оценить, для скольких точек $(a, b) \in S^{n-1} \times S$ выполняется $f(a, b) = 0$.

Доказательство (продолжение)

Доказательство

- Нам нужно оценить, для скольких точек $(a, b) \in S^{n-1} \times S$ выполняется $f(a, b) = 0$.
- Рассмотрим два случая:

Доказательство (продолжение)

Доказательство

- Нам нужно оценить, для скольких точек $(a, b) \in S^{n-1} \times S$ выполняется $f(a, b) = 0$.
- Рассмотрим два случая:
 - 1 $f_t(a) = 0$. Поскольку f_t — ненулевой многочлен полной степени не более $d - t$, то по индукционному предположению есть не более $(d - t)|S|^{n-2}$ точек в S^{n-1} , на которых он обращается в 0. Значит, всего есть не более $(d - t) \cdot |S|^{n-1}$ точек $(a, b) \in S^{n-1} \times S$, таких что $f(a, b) = 0$ и $f_t(a) = 0$.

Доказательство (продолжение)

Доказательство

- Нам нужно оценить, для скольких точек $(a, b) \in S^{n-1} \times S$ выполняется $f(a, b) = 0$.
- Рассмотрим два случая:
 - 1 $f_t(a) = 0$. Поскольку f_t — ненулевой многочлен полной степени не более $d - t$, то по индукционному предположению есть не более $(d - t)|S|^{n-2}$ точек в S^{n-1} , на которых он обращается в 0. Значит, всего есть не более $(d - t) \cdot |S|^{n-1}$ точек $(a, b) \in S^{n-1} \times S$, таких что $f(a, b) = 0$ и $f_t(a) = 0$.
 - 2 $f_t(a) \neq 0$. Для каждой точки $a \in S^{n-1}$, для которой $f_t(a) \neq 0$, $f(a, x_n)$ — ненулевой многочлен от одной переменной степени t . У такого многочлена не более t корней. Значит, всего есть не более $t \cdot |S|^{n-1}$ точек $(a, b) \in S^{n-1} \times S$, таких что $f(a, b) = 0$ и $f_t(a) \neq 0$.

Доказательство (продолжение)

Доказательство

- Нам нужно оценить, для скольких точек $(a, b) \in S^{n-1} \times S$ выполняется $f(a, b) = 0$.
- Рассмотрим два случая:
 - 1 $f_t(a) = 0$. Поскольку f_t — ненулевой многочлен полной степени не более $d - t$, то по индукционному предположению есть не более $(d - t)|S|^{n-2}$ точек в S^{n-1} , на которых он обращается в 0. Значит, всего есть не более $(d - t) \cdot |S|^{n-1}$ точек $(a, b) \in S^{n-1} \times S$, таких что $f(a, b) = 0$ и $f_t(a) = 0$.
 - 2 $f_t(a) \neq 0$. Для каждой точки $a \in S^{n-1}$, для которой $f_t(a) \neq 0$, $f(a, x_n)$ — ненулевой многочлен от одной переменной степени t . У такого многочлена не более t корней. Значит, всего есть не более $t \cdot |S|^{n-1}$ точек $(a, b) \in S^{n-1} \times S$, таких что $f(a, b) = 0$ и $f_t(a) \neq 0$.
- Таким образом, всего точек $(a, b) \in S^n$, на которых f обращается в 0, не более $(d - t) \cdot |S|^{n-1} + t \cdot |S|^{n-1} = d \cdot |S|^{n-1}$.



Слабая оценка

Слабая оценка

Слабая оценка

Слабая оценка

- Если f имеет степень d , тогда его полная степень может быть равна dn .

Слабая оценка

Слабая оценка

- Если f имеет степень d , тогда его полная степень может быть равна dn .
- Значит, даже для полилинейного многочлена оценка из только что доказанной леммы будет тривиальна, если $|S| \leq n$:

$$|S|^n - n \cdot |S|^{n-1} \leq 0.$$

Слабая оценка

Слабая оценка

- Если f имеет степень d , тогда его полная степень может быть равна dn .
- Значит, даже для полилинейного многочлена оценка из только что доказанной леммы будет тривиальна, если $|S| \leq n$:

$$|S|^n - n \cdot |S|^{n-1} \leq 0.$$

- Поэтому сейчас мы докажем другую лемму.

Еще одна лемма

Лемма

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен степени d над полем \mathbb{F} . Пусть также $S \subseteq \mathbb{F}$ и $|S| \geq d$. Тогда есть хотя бы $(|S| - d)^n$ точек $(a_1, \dots, a_n) \in S^n$, в которых $f(a_1, \dots, a_n) \neq 0$.

Еще одна лемма

Лемма

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен степени d над полем \mathbb{F} . Пусть также $S \subseteq \mathbb{F}$ и $|S| \geq d$. Тогда есть хотя бы $(|S| - d)^n$ точек $(a_1, \dots, a_n) \in S^n$, в которых $f(a_1, \dots, a_n) \neq 0$.

Доказательство

Еще одна лемма

Лемма

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен степени d над полем \mathbb{F} . Пусть также $S \subseteq \mathbb{F}$ и $|S| \geq d$. Тогда есть хотя бы $(|S| - d)^n$ точек $(a_1, \dots, a_n) \in S^n$, в которых $f(a_1, \dots, a_n) \neq 0$.

Доказательство

- Индукция по n .

Еще одна лемма

Лемма

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен степени d над полем \mathbb{F} . Пусть также $S \subseteq \mathbb{F}$ и $|S| \geq d$. Тогда есть хотя бы $(|S| - d)^n$ точек $(a_1, \dots, a_n) \in S^n$, в которых $f(a_1, \dots, a_n) \neq 0$.

Доказательство

- Индукция по n .
- База ($n = 1$) верна, поскольку многочлен от одной переменной степени d не может иметь более n корней.

Еще одна лемма

Лемма

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен степени d над полем \mathbb{F} . Пусть также $S \subseteq \mathbb{F}$ и $|S| \geq d$. Тогда есть хотя бы $(|S| - d)^n$ точек $(a_1, \dots, a_n) \in S^n$, в которых $f(a_1, \dots, a_n) \neq 0$.

Доказательство

- Индукция по n .
- База ($n = 1$) верна, поскольку многочлен от одной переменной степени d не может иметь более n корней.
- Рассмотрим точку $(a_1, \dots, a_n) \in S^n$, в которой $f(a_1, \dots, a_n) \neq 0$.

Еще одна лемма

Лемма

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен степени d над полем \mathbb{F} . Пусть также $S \subseteq \mathbb{F}$ и $|S| \geq d$. Тогда есть хотя бы $(|S| - d)^n$ точек $(a_1, \dots, a_n) \in S^n$, в которых $f(a_1, \dots, a_n) \neq 0$.

Доказательство

- Индукция по n .
- База ($n = 1$) верна, поскольку многочлен от одной переменной степени d не может иметь более n корней.
- Рассмотрим точку $(a_1, \dots, a_n) \in S^n$, в которой $f(a_1, \dots, a_n) \neq 0$.
- Рассмотрим следующие два многочлена:

$$\begin{aligned}f_0(x_1, \dots, x_{n-1}) &= f(x_1, \dots, x_{n-1}, a_n) \\f_1(x_n) &= f(a_1, \dots, a_{n-1}, x_n)\end{aligned}$$

Еще одна лемма

Лемма

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен степени d над полем \mathbb{F} . Пусть также $S \subseteq \mathbb{F}$ и $|S| \geq d$. Тогда есть хотя бы $(|S| - d)^n$ точек $(a_1, \dots, a_n) \in S^n$, в которых $f(a_1, \dots, a_n) \neq 0$.

Доказательство

- Индукция по n .
- База ($n = 1$) верна, поскольку многочлен от одной переменной степени d не может иметь более n корней.
- Рассмотрим точку $(a_1, \dots, a_n) \in S^n$, в которой $f(a_1, \dots, a_n) \neq 0$.
- Рассмотрим следующие два многочлена:

$$\begin{aligned}f_0(x_1, \dots, x_{n-1}) &= f(x_1, \dots, x_{n-1}, a_n) \\f_1(x_n) &= f(a_1, \dots, a_{n-1}, x_n)\end{aligned}$$

- Оба этих многочлена ненулевые и степени не более d .

Доказательство (продолжение)

Доказательство

Доказательство (продолжение)

Доказательство

- Многочлен f_0 зависит от $(n - 1)$ -й переменной, поэтому имеет хотя бы $(|S| - d)^{n-1}$ ненулевых точек в S^{n-1} (по индукционному предположению).

Доказательство (продолжение)

Доказательство

- Многочлен f_0 зависит от $(n - 1)$ -й переменной, поэтому имеет хотя бы $(|S| - d)^{n-1}$ ненулевых точек в S^{n-1} (по индукционному предположению).
- Аналогично многочлен f_1 зависит от одной переменной и, следовательно, имеет не менее $|S| - d$ ненулевых точек.

Доказательство (продолжение)

Доказательство

- Многочлен f_0 зависит от $(n - 1)$ -й переменной, поэтому имеет хотя бы $(|S| - d)^{n-1}$ ненулевых точек в S^{n-1} (по индукционному предположению).
- Аналогично многочлен f_1 зависит от одной переменной и, следовательно, имеет не менее $|S| - d$ ненулевых точек.
- Таким образом, для каждого из $|S| - d$ значений a_n , для которых $f_1 \neq 0$, соответствующий многочлен f_0 имеет хотя бы $(|S| - d)^{n-1}$ ненулевых точек.

Доказательство (продолжение)

Доказательство

- Многочлен f_0 зависит от $(n - 1)$ -й переменной, поэтому имеет хотя бы $(|S| - d)^{n-1}$ ненулевых точек в S^{n-1} (по индукционному предположению).
- Аналогично многочлен f_1 зависит от одной переменной и, следовательно, имеет не менее $|S| - d$ ненулевых точек.
- Таким образом, для каждого из $|S| - d$ значений a_n , для которых $f_1 \neq 0$, соответствующий многочлен f_0 имеет хотя бы $(|S| - d)^{n-1}$ ненулевых точек.
- Значит, всего ненулевых точек у многочлена f есть хотя бы $(|S| - d)^{n-1}(|S| - d) = (|S| - d)^n$.



Важное следствие

Следствие

Пусть $f(x_1, \dots, x_n)$ — ненулевой многочлен степени d от n переменных, $S \subseteq \mathbb{F}$ — конечное множество. Тогда для случайно выбранных x_1, \dots, x_n из S

$$\text{Prob}(f(x_1, \dots, x_n) \neq 0) \geq \left(1 - \frac{d}{|S|}\right)^n.$$

Алгоритм

Алгоритм

RAND-NULL-POLYNOMIAL(f)

Алгоритм

Алгоритм

RAND-NULL-POLYNOMIAL(f)

- Зафиксировать множество $S \subseteq \mathbb{F}$ размера $2dn$ (d — степень f , n — кол-во переменных).

Алгоритм

Алгоритм

RAND-NULL-POLYNOMIAL(f)

- Зафиксировать множество $S \subseteq \mathbb{F}$ размера $2dn$ (d — степень f , n — кол-во переменных).
- Повторить c раз:

Алгоритм

Алгоритм

RAND-NULL-POLYNOMIAL(f)

- Зафиксировать множество $S \subseteq \mathbb{F}$ размера $2dn$ (d — степень f , n — кол-во переменных).
- Повторить c раз:
 - ▶ случайно выбрать x_1, \dots, x_n из S

Алгоритм

Алгоритм

RAND-NULL-POLYNOMIAL(f)

- Зафиксировать множество $S \subseteq \mathbb{F}$ размера $2dn$ (d — степень f , n — кол-во переменных).
- Повторить c раз:
 - ▶ случайно выбрать x_1, \dots, x_n из S
 - ▶ если $f(x_1, \dots, x_n) \neq 0$, выдать “ненулевой”

Алгоритм

Алгоритм

RAND-NULL-POLYNOMIAL(f)

- Зафиксировать множество $S \subseteq \mathbb{F}$ размера $2dn$ (d — степень f , n — кол-во переменных).
- Повторить c раз:
 - ▶ случайно выбрать x_1, \dots, x_n из S
 - ▶ если $f(x_1, \dots, x_n) \neq 0$, выдать “ненулевой”
- выдать “нулевой”

Анализ алгоритма

Лемма

Алгоритм `RAND-NULL-POLYNOMIAL` ошибается с вероятностью не более 2^{-c} , причем только в случае, если f — ненулевой.

Доказательство

Анализ алгоритма

Лемма

Алгоритм `RAND-NULL-POLYNOMIAL` ошибается с вероятностью не более 2^{-c} , причем только в случае, если f — ненулевой.

Доказательство

- $1 - t > e^{-t-t^2}$, $0 < t \leq 1/2$

Анализ алгоритма

Лемма

Алгоритм `RAND-NULL-POLYNOMIAL` ошибается с вероятностью не более 2^{-c} , причем только в случае, если f — ненулевой.

Доказательство

- $1 - t > e^{-t-t^2}$, $0 < t \leq 1/2$
- $1 - \frac{1}{2n} > e^{-\frac{1}{2n} - \frac{1}{4n^2}}$

Анализ алгоритма

Лемма

Алгоритм RAND-NULL-POLYNOMIAL ошибается с вероятностью не более 2^{-c} , причем только в случае, если f — ненулевой.

Доказательство

- $1 - t > e^{-t-t^2}$, $0 < t \leq 1/2$
- $1 - \frac{1}{2n} > e^{-\frac{1}{2n} - \frac{1}{4n^2}}$
- Случайно выбранный набор значений переменным x_1, \dots, x_n является корнем с вероятностью не более

$$1 - \left(1 - \frac{1}{2n}\right)^n < 1 - e^{-\frac{1}{2} - \frac{1}{4n}} < \frac{1}{2}.$$



План лекции

- 1 Проверка перемножения матриц
- 2 Сравнение строк
- 3 Нулевой многочлен
- 4 Эквивалентность ветвящихся программ

Пример

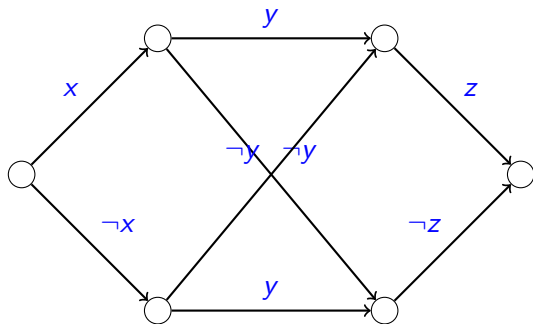


Рис.: Ветвящаяся программа степени 1, вычисляющая функцию четности $x \oplus y \oplus z$.

Ветвящиеся программы

Определение

Ветвящаяся программа (branching program) представляет собой ориентированный граф $P = (V, E)$ со следующими свойствами:

Ветвящиеся программы

Определение

Ветвящаяся программа (branching program) представляет собой ориентированный граф $P = (V, E)$ со следующими свойствами:

- есть две выделенные вершины — начальная и конечная;

Ветвящиеся программы

Определение

Ветвящаяся программа (branching program) представляет собой ориентированный граф $P = (V, E)$ со следующими свойствами:

- есть две выделенные вершины — начальная и конечная;
- исходящая степень каждой вершины равна 0, 1 или 2;

Ветвящиеся программы

Определение

Ветвящаяся программа (branching program) представляет собой ориентированный граф $P = (V, E)$ со следующими свойствами:

- есть две выделенные вершины — начальная и конечная;
- исходящая степень каждой вершины равна 0, 1 или 2;
- каждое ребро помечено переменной x_i или ее отрицанием $\neg x_i$;
более того, если из вершины выходят два ребра, то они обязательно помечены x_i и $\neg x_i$;

Функция

Определение

Определение

- **Мономом** (product term) называется конъюнкция всех пометок на пути, ведущем из начальной в конечную вершину (если такой путь содержит одновременно и переменную, и ее отрицание, то моном равен 0).

Определение

- **Мономом** (product term) называется конъюнкция всех пометок на пути, ведущем из начальной в конечную вершину (если такой путь содержит одновременно и переменную, и ее отрицание, то моном равен 0).
- Функция f_P , вычисляемая программой P , определяется как дизъюнкция всех мономов.

Определение

- **Мономом** (product term) называется конъюнкция всех пометок на пути, ведущем из начальной в конечную вершину (если такой путь содержит одновременно и переменную, и ее отрицание, то моном равен 0).
- Функция f_P , вычисляемая программой P , определяется как дизъюнкция всех мономов.
- Ветвящаяся программа имеет **первую степень** (read-once), если никакая переменная не встречается ни на каком пути более одного раза.

Эквивалентность программ

Эквивалентность программ

Эквивалентность программ

Эквивалентность программ

- Мы хотим проверить по данным двум ветвящимся программам P и Q , эквивалентны ли они.

Эквивалентность программ

Эквивалентность программ

- Мы хотим проверить по данным двум ветвящимся программам P и Q , эквивалентны ли они.
- Программе P можно сопоставить многочлен f_P (над любым полем \mathbb{F} , содержащим хотя бы два элемента).

Эквивалентность программ

Эквивалентность программ

- Мы хотим проверить по данным двум ветвящимся программам P и Q , эквивалентны ли они.
- Программе P можно сопоставить многочлен f_P (над любым полем \mathbb{F} , содержащим хотя бы два элемента).
 - ▶ Для этого каждому ребру e поставим в соответствие многочлен $f_e = x_i$ или $f_e = 1 - x_i$, если оно помечено переменной x_i или ее отрицанием $\neg x_i$, соответственно.

Эквивалентность программ

Эквивалентность программ

- Мы хотим проверить по данным двум ветвящимся программам P и Q , эквивалентны ли они.
- Программе P можно сопоставить многочлен f_P (над любым полем \mathbb{F} , содержащим хотя бы два элемента).
 - ▶ Для этого каждому ребру e поставим в соответствие многочлен $f_e = x_i$ или $f_e = 1 - x_i$, если оно помечено переменной x_i или ее отрицанием $\neg x_i$, соответственно.
 - ▶ Каждому пути поставим в соответствие произведение многочленов ребер этого пути.

Эквивалентность программ

Эквивалентность программ

- Мы хотим проверить по данным двум ветвящимся программам P и Q , эквивалентны ли они.
- Программе P можно сопоставить многочлен f_P (над любым полем \mathbb{F} , содержащим хотя бы два элемента).
 - ▶ Для этого каждому ребру e поставим в соответствие многочлен $f_e = x_i$ или $f_e = 1 - x_i$, если оно помечено переменной x_i или ее отрицанием $\neg x_i$, соответственно.
 - ▶ Каждому пути поставим в соответствие произведение многочленов ребер этого пути.
 - ▶ Программе сопоставим многочлен, равный сумме многочленов всех ее путей.

Эквивалентность программ

Эквивалентность программ

- Мы хотим проверить по данным двум ветвящимся программам P и Q , эквивалентны ли они.
- Программе P можно сопоставить многочлен f_P (над любым полем \mathbb{F} , содержащим хотя бы два элемента).
 - ▶ Для этого каждому ребру e поставим в соответствие многочлен $f_e = x_i$ или $f_e = 1 - x_i$, если оно помечено переменной x_i или ее отрицанием $\neg x_i$, соответственно.
 - ▶ Каждому пути поставим в соответствие произведение многочленов ребер этого пути.
 - ▶ Программе сопоставим многочлен, равный сумме многочленов всех ее путей.
- Полученный многочлен вычисляет функцию из \mathbb{F}^n в \mathbb{F} .

Эквивалентность программ

Эквивалентность программ

- Мы хотим проверить по данным двум ветвящимся программам P и Q , эквивалентны ли они.
- Программе P можно сопоставить многочлен f_P (над любым полем \mathbb{F} , содержащим хотя бы два элемента).
 - ▶ Для этого каждому ребру e поставим в соответствие многочлен $f_e = x_i$ или $f_e = 1 - x_i$, если оно помечено переменной x_i или ее отрицанием $\neg x_i$, соответственно.
 - ▶ Каждому пути поставим в соответствие произведение многочленов ребер этого пути.
 - ▶ Программе сопоставим многочлен, равный сумме многочленов всех ее путей.
- Полученный многочлен вычисляет функцию из \mathbb{F}^n в \mathbb{F} .
- Ясно, что f_P и P эквивалентны.

Эквивалентность многочленов

Эквивалентность многочленов

Эквивалентность многочленов

Эквивалентность многочленов

- Допустим теперь, что ветвящиеся программы P и Q являются программами степени 1.

Эквивалентность многочленов

Эквивалентность многочленов

- Допустим теперь, что ветвящиеся программы P и Q являются программами степени 1.
- Тогда многочлен $f_P - f_Q$ полилинеен.

Эквивалентность многочленов

Эквивалентность многочленов

- Допустим теперь, что ветвящиеся программы P и Q являются программами степени 1 .
- Тогда многочлен $f_P - f_Q$ полилинеен.
- Если выбрать $S \subseteq \mathbb{F}$, $|S| \geq 2n$, то в предположении, что $f_P - f_Q \neq 0$, для случайно выбранных r_1, \dots, r_n

$$\text{Prob}(f_P(r_1, \dots, r_n) = f_Q(r_1, \dots, r_n)) \leq 1 - \left(1 - \frac{1}{|S|}\right)^n < 1/2.$$

Однако

Однако

Однако

Однако

- Мы должны научиться быстро вычислять значение многочлена f_p на любом входе.

Однако

Однако

- Мы должны научиться быстро вычислять значение многочлена f_p на любом входе.
- Размер входа исходной задачи — это количество вершин в графе программы. Размер соответствующего многочлена может быть экспоненциален от размера входа.

Однако

Однако

- Мы должны научиться быстро вычислять значение многочлена f_P на любом входе.
- Размер входа исходной задачи — это количество вершин в графе программы. Размер соответствующего многочлена может быть экспоненциален от размера входа.
- Для входа $(r_1, \dots, r_n) \in \mathbb{F}^n$ мы можем вычислить f_P следующим образом:

Однако

Однако

- Мы должны научиться быстро вычислять значение многочлена f_P на любом входе.
- Размер входа исходной задачи — это количество вершин в графе программы. Размер соответствующего многочлена может быть экспоненциален от размера входа.
- Для входа $(r_1, \dots, r_n) \in \mathbb{F}^n$ мы можем вычислить f_P следующим образом:
 - ▶ присвоим начальной вершине значение 1 ;

Однако

Однако

- Мы должны научиться быстро вычислять значение многочлена f_P на любом входе.
- Размер входа исходной задачи — это количество вершин в графе программы. Размер соответствующего многочлена может быть экспоненциален от размера входа.
- Для входа $(r_1, \dots, r_n) \in \mathbb{F}^n$ мы можем вычислить f_P следующим образом:
 - ▶ присвоим начальной вершине значение 1 ;
 - ▶ каждому ребру присвоим значение его многочлена (то есть каждому ребру будет присвоено значение r_i или $1 - r_i$);

Однако

Однако

- Мы должны научиться быстро вычислять значение многочлена f_P на любом входе.
- Размер входа исходной задачи — это количество вершин в графе программы. Размер соответствующего многочлена может быть экспоненциален от размера входа.
- Для входа $(r_1, \dots, r_n) \in \mathbb{F}^n$ мы можем вычислить f_P следующим образом:
 - ▶ присвоим начальной вершине значение 1 ;
 - ▶ каждому ребру присвоим значение его многочлена (то есть каждому ребру будет присвоено значение r_i или $1 - r_i$);
 - ▶ каждой вершине присвоим сумму по всем входящим в нее ребрам произведений значения ребра на значение ребра на другом конце.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

проверка результата перемножения матриц (домножить на случайный булев вектор)

Что мы узнали за сегодня?

Что мы узнали за сегодня?

проверка результата перемножения матриц (домножить на случайный булев вектор)

сравнение строк (представить строки как числа и проверять равенство по случайно выбранному простому модулю)

Что мы узнали за сегодня?

Что мы узнали за сегодня?

проверка результата перемножения матриц (домножить на случайный булев вектор)

сравнение строк (представить строки как числа и проверять равенство по случайно выбранному простому модулю)

нулевой многочлен (взять случайную точку и посчитать в ней значение)

Что мы узнали за сегодня?

Что мы узнали за сегодня?

проверка результата перемножения матриц (домножить на случайный булев вектор)

сравнение строк (представить строки как числа и проверять равенство по случайно выбранному простому модулю)

нулевой многочлен (взять случайную точку и посчитать в ней значение)

эквивалентность ветвящихся программ (сопоставить программе многочлен и воспользоваться предыдущим алгоритмом)

Спасибо за внимание!