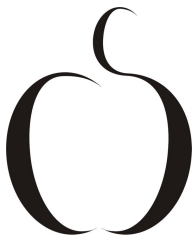


с/к “Эффективные алгоритмы”

Лекция 10: Вероятностные алгоритмы

А. Куликов

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



План лекции

1 Минимальный разрез

План лекции

- 1 Минимальный разрез
- 2 Проверка простоты числа

План лекции

- 1 Минимальный разрез
- 2 Проверка простоты числа

Минимальный разрез

Определение

Задача о минимальном разрезе (min-cut problem) заключается в нахождении по данному связному мультиграфу такого минимального количества ребер, удаление которых делает граф несвязным.

Минимальный разрез

Определение

Задача о минимальном разрезе (min-cut problem) заключается в нахождении по данному связному мультиграфу такого минимального количества ребер, удаление которых делает граф несвязным.

Алгоритм

RAND-MIN-CUT(G)

Минимальный разрез

Определение

Задача о минимальном разрезе (min-cut problem) заключается в нахождении по данному связному мультиграфу такого минимального количества ребер, удаление которых делает граф несвязным.

Алгоритм

RAND-MIN-CUT(G)

- повторять, пока в G более двух вершин:

Минимальный разрез

Определение

Задача о минимальном разрезе (min-cut problem) заключается в нахождении по данному связному мультиграфу такого минимального количества ребер, удаление которых делает граф несвязным.

Алгоритм

RAND-MIN-CUT(G)

- повторять, пока в G более двух вершин:
 - ▶ выбрать случайное ребро (u, v)

Минимальный разрез

Определение

Задача о минимальном разрезе (min-cut problem) заключается в нахождении по данному связному мультиграфу такого минимального количества ребер, удаление которых делает граф несвязным.

Алгоритм

RAND-MIN-CUT(G)

- повторять, пока в G более двух вершин:
 - ▶ выбрать случайное ребро (u, v)
 - ▶ удалить все ребра между u и v

Минимальный разрез

Определение

Задача о минимальном разрезе (min-cut problem) заключается в нахождении по данному связному мультиграфу такого минимального количества ребер, удаление которых делает граф несвязным.

Алгоритм

RAND-MIN-CUT(G)

- повторять, пока в G более двух вершин:
 - ▶ выбрать случайное ребро (u, v)
 - ▶ удалить все ребра между u и v
 - ▶ стянуть u и v в одну

Минимальный разрез

Определение

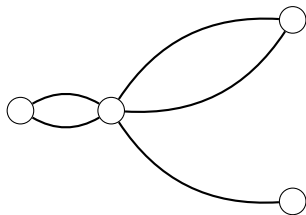
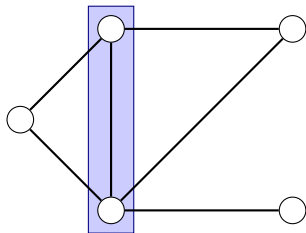
Задача о минимальном разрезе (min-cut problem) заключается в нахождении по данному связному мультиграфу такого минимального количества ребер, удаление которых делает граф несвязным.

Алгоритм

RAND-MIN-CUT(G)

- повторять, пока в G более двух вершин:
 - ▶ выбрать случайное ребро (u, v)
 - ▶ удалить все ребра между u и v
 - ▶ стянуть u и v в одну
- вернуть количество ребер между двумя оставшимися вершинами

Пример стягивания



Анализ алгоритма

Анализ алгоритма

Анализ алгоритма

Анализ алгоритма

- Заметим, что стягивание не уменьшает размера минимального разреза: любой разрез в графе со стянутыми вершинами является разрезом и в исходном графе.

Анализ алгоритма

Анализ алгоритма

- Заметим, что стягивание не уменьшает размера минимального разреза: любой разрез в графе со стянутыми вершинами является разрезом и в исходном графе.
- Если минимальный размер разреза графа G равен k , то в графе G хотя бы $kn/2$ ребер: степень каждой вершины хотя бы k .

Анализ алгоритма

Анализ алгоритма

- Заметим, что стягивание не уменьшает размера минимального разреза: любой разрез в графе со стянутыми вершинами является разрезом и в исходном графе.
- Если минимальный размер разреза графа G равен k , то в графе G хотя бы $kn/2$ ребер: степень каждой вершины хотя бы k .
- Пусть C — минимальный разрез графа размера k .

Анализ алгоритма

Анализ алгоритма

- Заметим, что стягивание не уменьшает размера минимального разреза: любой разрез в графе со стянутыми вершинами является разрезом и в исходном графе.
- Если минимальный размер разреза графа G равен k , то в графе G хотя бы $kn/2$ ребер: степень каждой вершины хотя бы k .
- Пусть C — минимальный разрез графа размера k .
- Обозначим событие “ребро из C не было выбрано на шаге i ” через A_i .

Анализ алгоритма

Анализ алгоритма

- Заметим, что стягивание не уменьшает размера минимального разреза: любой разрез в графе со стянутыми вершинами является разрезом и в исходном графе.
- Если минимальный размер разреза графа G равен k , то в графе G хотя бы $kn/2$ ребер: степень каждой вершины хотя бы k .
- Пусть C — минимальный разрез графа размера k .
- Обозначим событие “ребро из C не было выбрано на шаге i ” через A_i .
- Тогда $\mathbf{Prob}(\text{алгоритм выдаст } C) = \mathbf{Prob}(A_1 \cap A_2 \cap \dots \cap A_{n-2})$.

Анализ алгоритма (продолжение)

Анализ алгоритма

Анализ алгоритма (продолжение)

Анализ алгоритма

- $\text{Prob}(A_1) \geq 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n}$

Анализ алгоритма (продолжение)

Анализ алгоритма

- $\text{Prob}(A_1) \geq 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n}$
- После первого шага остается $n - 1$ вершин и хотя бы $k(n - 1)/2$ ребер.

Анализ алгоритма (продолжение)

Анализ алгоритма

- $\text{Prob}(A_1) \geq 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n}$
- После первого шага остается $n - 1$ вершин и хотя бы $k(n - 1)/2$ ребер.
- $\text{Prob}(A_2|A_1) \geq 1 - \frac{2}{n-1}$

Анализ алгоритма (продолжение)

Анализ алгоритма

- $\text{Prob}(A_1) \geq 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n}$
- После первого шага остается $n - 1$ вершин и хотя бы $k(n - 1)/2$ ребер.
- $\text{Prob}(A_2|A_1) \geq 1 - \frac{2}{n-1}$
- На i -ом шаге в графе есть $n - (i - 1)$ вершин, размер минимального разреза хотя бы k , поэтому ребер в графе хотя бы $k(n - i + 1)/2$.

Анализ алгоритма (продолжение)

Анализ алгоритма

- $\text{Prob}(A_1) \geq 1 - \frac{|C|}{|E|} \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n}$
- После первого шага остается $n - 1$ вершин и хотя бы $k(n - 1)/2$ ребер.
- $\text{Prob}(A_2|A_1) \geq 1 - \frac{2}{n-1}$
- На i -ом шаге в графе есть $n - (i - 1)$ вершин, размер минимального разреза хотя бы k , поэтому ребер в графе хотя бы $k(n - i + 1)/2$.
-

$$\text{Prob} \left(A_i \mid \bigcap_{j=1}^{i-1} A_j \right) \geq 1 - \frac{2}{n - i + 1}$$

Анализ алгоритма (продолжение)

Анализ алгоритма

Анализ алгоритма (продолжение)

Анализ алгоритма



$$\text{Prob} \left(\bigcap_{j=1}^{n-2} A_j \right) = \text{Prob} (A_1) \cdot \text{Prob} (A_2 | A_1) \cdot \dots \cdot \text{Prob} \left(A_{n-2} | \bigcap_{j=1}^{n-3} A_j \right)$$

Анализ алгоритма (продолжение)

Анализ алгоритма



$$\text{Prob} \left(\bigcap_{j=1}^{n-2} A_j \right) = \text{Prob}(A_1) \cdot \text{Prob}(A_2|A_1) \cdot \dots \cdot \text{Prob} \left(A_{n-2} \mid \bigcap_{j=1}^{n-3} A_j \right)$$

- Итак, алгоритм выдает разрез C с вероятностью хотя бы

$$\prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1} \right) = \frac{2}{n(n-1)}.$$

Анализ алгоритма (продолжение)

Лемма

Алгоритм RAND-MIN-CUT выдает каждое минимальное сечение графа с вероятностью хотя бы $\frac{2}{n(n-1)}$. Время работы алгоритма — $O(n^2)$.

Улучшение алгоритма

Улучшение алгоритма

Улучшение алгоритма

Улучшение алгоритма

- Таким образом, чтобы получить константную вероятность ошибки, придется запустить этот алгоритм $O(n^2)$ раз, что приведет ко времени работы $O(n^4)$.

Улучшение алгоритма

Улучшение алгоритма

- Таким образом, чтобы получить константную вероятность ошибки, придется запустить этот алгоритм $O(n^2)$ раз, что приведет ко времени работы $O(n^4)$.
- Как можно улучшить алгоритм?

Улучшение алгоритма

Улучшение алгоритма

- Таким образом, чтобы получить константную вероятность ошибки, придется запустить этот алгоритм $O(n^2)$ раз, что приведет ко времени работы $O(n^4)$.
- Как можно улучшить алгоритм?
- Как мы уже видели, на i -ом шаге алгоритм не выбирает ребро из зафиксированного покрытия с вероятностью $1 - \frac{2}{n-i+1}$.

Улучшение алгоритма

Улучшение алгоритма

- Таким образом, чтобы получить константную вероятность ошибки, придется запустить этот алгоритм $O(n^2)$ раз, что приведет ко времени работы $O(n^4)$.
- Как можно улучшить алгоритм?
- Как мы уже видели, на i -ом шаге алгоритм не выбирает ребро из зафиксированного покрытия с вероятностью $1 - \frac{2}{n-i+1}$.
- С увеличением i данная вероятность убывает.

Улучшение алгоритма

Улучшение алгоритма

- Таким образом, чтобы получить константную вероятность ошибки, придется запустить этот алгоритм $O(n^2)$ раз, что приведет ко времени работы $O(n^4)$.
- Как можно улучшить алгоритм?
- Как мы уже видели, на i -ом шаге алгоритм не выбирает ребро из зафиксированного покрытия с вероятностью $1 - \frac{2}{n-i+1}$.
- С увеличением i данная вероятность убывает.
- Идея: будем стягивать вершины, только пока соответствующая вероятность достаточно хороша.

Улучшенный алгоритм

Алгоритм

IMPROVED-RAND-MIN-CUT(G)

Улучшенный алгоритм

Алгоритм

IMPROVED-RAND-MIN-CUT(G)

- $t = n/\sqrt{2}$

Улучшенный алгоритм

Алгоритм

IMPROVED-RAND-MIN-CUT(G)

- $t = n/\sqrt{2}$
- стягиванием случайных ребер получить два графа H_1 и H_2 на t вершинах

Улучшенный алгоритм

Алгоритм

IMPROVED-RAND-MIN-CUT(G)

- $t = n/\sqrt{2}$
- стягиванием случайных ребер получить два графа H_1 и H_2 на t вершинах
- $k_1 = \text{IMPROVED-RAND-MIN-CUT}(H_1)$

Улучшенный алгоритм

Алгоритм

IMPROVED-RAND-MIN-CUT(G)

- $t = n/\sqrt{2}$
- стягиванием случайных ребер получить два графа H_1 и H_2 на t вершинах
- $k_1 = \text{IMPROVED-RAND-MIN-CUT}(H_1)$
- $k_2 = \text{IMPROVED-RAND-MIN-CUT}(H_2)$

Улучшенный алгоритм

Алгоритм

IMPROVED-RAND-MIN-CUT(G)

- $t = n/\sqrt{2}$
- стягиванием случайных ребер получить два графа H_1 и H_2 на t вершинах
- $k_1 = \text{IMPROVED-RAND-MIN-CUT}(H_1)$
- $k_2 = \text{IMPROVED-RAND-MIN-CUT}(H_2)$
- вернуть $\min\{k_1, k_2\}$

Анализ времени работы

Лемма

Время работы алгоритма IMPROVED-RAND-MIN-CUT — $O(n^2 \log n)$.

Анализ времени работы

Лемма

Время работы алгоритма IMPROVED-RAND-MIN-CUT — $O(n^2 \log n)$.

Доказательство

Анализ времени работы

Лемма

Время работы алгоритма IMPROVED-RAND-MIN-CUT — $O(n^2 \log n)$.

Доказательство



$$T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + O(n^2)$$

Анализ времени работы

Лемма

Время работы алгоритма IMPROVED-RAND-MIN-CUT — $O(n^2 \log n)$.

Доказательство

- $$T(n) = 2T\left(\frac{n}{\sqrt{2}}\right) + O(n^2)$$
- $T(n) = O(n^2 \log n)$



Оценка вероятности ошибки

Лемма

Алгоритм IMPROVED-RAND-MIN-CUT выдает минимальный разрез графа с вероятностью $\Omega(1/\log n)$.

Оценка вероятности ошибки

Лемма

Алгоритм IMPROVED-RAND-MIN-CUT выдает минимальный разрез графа с вероятностью $\Omega(1/\log n)$.

Доказательство

Оценка вероятности ошибки

Лемма

Алгоритм IMPROVED-RAND-MIN-CUT выдает минимальный разрез графа с вероятностью $\Omega(1/\log n)$.

Доказательство

- Допустим, разрез размера k все еще существует в графе H на t вершинах.

Оценка вероятности ошибки

Лемма

Алгоритм IMPROVED-RAND-MIN-CUT выдает минимальный разрез графа с вероятностью $\Omega(1/\log n)$.

Доказательство

- Допустим, разрез размера k все еще существует в графе H на t вершинах.
- Алгоритм вернет этот разрез, если для какого-то из двух графов H_1 и H_2 ни одно ребро этого разреза не будет стянуто и соответствующий рекурсивный вызов вернет именно этот разрез.

Оценка вероятности ошибки

Лемма

Алгоритм IMPROVED-RAND-MIN-CUT выдает минимальный разрез графа с вероятностью $\Omega(1/\log n)$.

Доказательство

- Допустим, разрез размера k все еще существует в графе H на t вершинах.
- Алгоритм вернет этот разрез, если для какого-то из двух графов H_1 и H_2 ни одно ребро этого разреза не будет стянуто и соответствующий рекурсивный вызов вернет именно этот разрез.
- При стягивании t вершин в $t/\sqrt{2}$ вероятность того, что ни одно ребро разреза стянуто не будет, хотя бы

$$\frac{t/\sqrt{2}(t/\sqrt{2} - 1)}{t(t - 1)} \geq \frac{1}{2}.$$

Доказательство (продолжение)

Доказательство

Доказательство (продолжение)

Доказательство

- Обозначим через $P(t)$ вероятность того, что алгоритм находит минимальный разрез графа на t вершинах.

Доказательство (продолжение)

Доказательство

- Обозначим через $P(t)$ вероятность того, что алгоритм находит минимальный разрез графа на t вершинах.



$$P(t) \geq 1 - \left(1 - \frac{1}{2}P\left(\frac{t}{\sqrt{2}}\right)\right)^2$$

Доказательство (продолжение)

Доказательство

- Обозначим через $P(t)$ вероятность того, что алгоритм находит минимальный разрез графа на t вершинах.



$$P(t) \geq 1 - \left(1 - \frac{1}{2}P\left(\frac{t}{\sqrt{2}}\right)\right)^2$$

- Нетрудно проверить, что $P(t) = \Theta\left(\frac{1}{t}\right)$ является решением.



План лекции

- 1 Минимальный разрез
- 2 Проверка простоты числа

Проверка простоты числа

Проверка простоты числа

Проверка простоты числа

Проверка простоты числа

- Как найти большое простое число?

Проверка простоты числа

Проверка простоты числа

- Как найти большое простое число?
- Можно, например, выбрать случайное число n и проверить, является ли оно простым.

Проверка простоты числа

Проверка простоты числа

- Как найти большое простое число?
- Можно, например, выбрать случайное число n и проверить, является ли оно простым.
- Но как эффективно проверять?

Проверка простоты числа

Проверка простоты числа

- Как найти большое простое число?
- Можно, например, выбрать случайное число n и проверить, является ли оно простым.
- Но как эффективно проверять?
- Можно перебрать все возможные делители от 2 до \sqrt{n} , но это займет экспоненциальное время от размера входа: $\sqrt{n} = 2^{\log_2 n/2}$.

Проверка простоты числа

Проверка простоты числа

- Как найти большое простое число?
- Можно, например, выбрать случайное число n и проверить, является ли оно простым.
- Но как эффективно проверять?
- Можно перебрать все возможные делители от 2 до \sqrt{n} , но это займет экспоненциальное время от размера входа: $\sqrt{n} = 2^{\log_2 n/2}$.
- Мы предъявим полиномиальный вероятностный алгоритм.

Факты

Факты

Факты

Факты

- Известен детерминированный полиномиальный алгоритм проверки числа на простоту.

Факты

Факты

- Известен детерминированный полиномиальный алгоритм проверки числа на простоту.
- Однако нет ни одного известного полиномиального алгоритма разложения числа на множители.

Факты

- Известен детерминированный полиномиальный алгоритм проверки числа на простоту.
- Однако нет ни одного известного полиномиального алгоритма разложения числа на множители.
- Самое больше известное (на данный момент) простое число:

$$33\,661 \cdot 2^{7\,031\,232} + 1.$$

Факты

- Известен детерминированный полиномиальный алгоритм проверки числа на простоту.
- Однако нет ни одного известного полиномиального алгоритма разложения числа на множители.
- Самое больше известное (на данный момент) простое число:

$$33\,661 \cdot 2^{7\,031\,232} + 1.$$

- В его десятичной записи **2116617** цифр.

Малая теорема Ферма

Определение

Малая теорема Ферма

Определение

- $\mathbb{Z}_n^+ = \{1, 2, \dots, n - 1\}$ — множество всех ненулевых вычетов (остатков) по модулю n .

Малая теорема Ферма

Определение

- $\mathbb{Z}_n^+ = \{1, 2, \dots, n-1\}$ — множество всех ненулевых вычетов (остатков) по модулю n .
- \mathbb{Z}_n^* — множество остатков, взаимно простых с n (для простого n $\mathbb{Z}_n^+ = \mathbb{Z}_n^*$).

Малая теорема Ферма

Определение

- $\mathbb{Z}_n^+ = \{1, 2, \dots, n-1\}$ — множество всех ненулевых вычетов (остатков) по модулю n .
- \mathbb{Z}_n^* — множество остатков, взаимно простых с n (для простого n $\mathbb{Z}_n^+ = \mathbb{Z}_n^*$).

Теорема

Если $p \in \mathbb{P}$ и $(a, p) = 1$, то

$$a^{p-1} \equiv 1 \pmod{p}.$$

Малая теорема Ферма

Определение

- $\mathbb{Z}_n^+ = \{1, 2, \dots, n-1\}$ — множество всех ненулевых вычетов (остатков) по модулю n .
- \mathbb{Z}_n^* — множество остатков, взаимно простых с n (для простого n $\mathbb{Z}_n^+ = \mathbb{Z}_n^*$).

Теорема

Если $p \in \mathbb{P}$ и $(a, p) = 1$, то

$$a^{p-1} \equiv 1 \pmod{p}.$$

Определение

Число n называется **псевдопростым по основанию a** (base- a pseudoprime), если $a^{n-1} \equiv 1 \pmod{n}$.

Простой тест

Простой тест

PSEUDOPRIME(n)

Простой тест

Простой тест

PSEUDOPRIME(n)

- если $2^{n-1} \not\equiv 1 \pmod n$, вернуть “составное”

Простой тест

Простой тест

PSEUDOPRIME(n)

- если $2^{n-1} \not\equiv 1 \pmod n$, вернуть “составное”
- вернуть “простое”

Простой тест

Простой тест

PSEUDOPRIME(n)

- если $2^{n-1} \not\equiv 1 \pmod n$, вернуть “составное”
- вернуть “простое”

Анализ

Простой тест

Простой тест

PSEUDOPRIME(n)

- если $2^{n-1} \not\equiv 1 \pmod n$, вернуть “составное”
- вернуть “простое”

Анализ

- Существуют составные числа, которые данный тест не распознает.

Простой тест

Простой тест

PSEUDOPRIME(n)

- если $2^{n-1} \not\equiv 1 \pmod n$, вернуть “составное”
- вернуть “простое”

Анализ

- Существуют составные числа, которые данный тест не распознает.
- Таких чисел довольно мало, но все же.

Простой тест

Простой тест

PSEUDOPRIME(n)

- если $2^{n-1} \not\equiv 1 \pmod n$, вернуть “составное”
- вернуть “простое”

Анализ

- Существуют составные числа, которые данный тест не распознает.
- Таких чисел довольно мало, но все же.
- Можно проверять другие основания, но и это не поможет: существуют числа n , псевдопростые по любому основанию $a \in \mathbb{Z}_n^*$.

Улучшение теста

Улучшение теста

Улучшение теста

Улучшение теста

- Будем проверять несколько оснований.

Улучшение теста

Улучшение теста

- Будем проверять несколько оснований.
- Вычисляя a^{n-1} , будем проверять, не нашлось ли такого $x \leq n$, что $x^2 \equiv 1 \pmod n$:

Улучшение теста

Улучшение теста

- Будем проверять несколько оснований.
- Вычисляя a^{n-1} , будем проверять, не нашлось ли такого $x \leq n$, что $x^2 \equiv 1 \pmod n$:
 - ▶ если $x^2 \equiv 1 \pmod n$, то $(x-1)(x+1) \equiv 0 \pmod n$, тогда если $n \in \mathbb{P}$, то x равен 1 или $n-1$.

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$
- $d = 1$

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$
- $d = 1$
- для i от k до 0 повторять:

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$
- $d = 1$
- для i от k до 0 повторять:
 - ▶ $x = d$

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$
- $d = 1$
- для i от k до 0 повторять:
 - ▶ $x = d$
 - ▶ $d = d^2 \pmod n$

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$
- $d = 1$
- для i от k до 0 повторять:
 - ▶ $x = d$
 - ▶ $d = d^2 \pmod n$
 - ▶ если $d = 1$ и $x \neq 1$ и $x \neq n - 1$, вернуть “составное”

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$
- $d = 1$
- для i от k до 0 повторять:
 - ▶ $x = d$
 - ▶ $d = d^2 \pmod n$
 - ▶ если $d = 1$ и $x \neq 1$ и $x \neq n - 1$, вернуть “составное”
 - ▶ если $b_i = 1$, $d = (d \cdot a) \pmod n$

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$
- $d = 1$
- для i от k до 0 повторять:
 - ▶ $x = d$
 - ▶ $d = d^2 \bmod n$
 - ▶ если $d = 1$ и $x \neq 1$ и $x \neq n - 1$, вернуть “составное”
 - ▶ если $b_i = 1$, $d = (d \cdot a) \bmod n$
- если $d \neq 1$, вернуть “составное”

Возведение в степень с прокеркой

Возведение в степень с проверкой

WITNESS(a, n)

- пусть $\langle b_k, b_{k-1}, \dots, b_0 \rangle$ — двоичная запись $n - 1$
- $d = 1$
- для i от k до 0 повторять:
 - ▶ $x = d$
 - ▶ $d = d^2 \pmod n$
 - ▶ если $d = 1$ и $x \neq 1$ и $x \neq n - 1$, вернуть “составное”
 - ▶ если $b_i = 1$, $d = (d \cdot a) \pmod n$
- если $d \neq 1$, вернуть “составное”
- вернуть “простое”

Алгоритм

Алгоритм

PRIME(n, s)

Алгоритм

Алгоритм

PRIME(n, s)

- повторить s раз

Алгоритм

Алгоритм

PRIME(n, s)

- повторить s раз
 - ▶ выбрать случайное основание a от 1 до $n - 1$

Алгоритм

Алгоритм

PRIME(n, s)

- повторить s раз
 - ▶ выбрать случайное основание a от 1 до $n - 1$
 - ▶ если WITNESS(a, n) вернул “составное”, вернуть “составное”

Алгоритм

Алгоритм

PRIME(n, s)

- повторить s раз
 - ▶ выбрать случайное основание a от 1 до $n - 1$
 - ▶ если WITNESS(a, n) вернул “составное”, вернуть “составное”
- вернуть “простое”

Алгоритм

Алгоритм

PRIME(n, s)

- повторить s раз
 - ▶ выбрать случайное основание a от 1 до $n - 1$
 - ▶ если WITNESS(a, n) вернул “составное”, вернуть “составное”
- вернуть “простое”

Время работы алгоритма

Возведение в степень требует $\log n$ операций.

Основная теорема

Теорема

Для нечетного составного числа n

$$|\{a \in \mathbb{Z}_n^+ : \text{WITNESS}(a, n) = \text{“составное”}\}| \geq \frac{n-1}{2}.$$

Идея доказательства

Основная теорема

Теорема

Для нечетного составного числа n

$$|\{a \in \mathbb{Z}_n^+ : \text{WITNESS}(a, n) = \text{“составное”}\}| \geq \frac{n-1}{2}.$$

Идея доказательства

- Назовем основание $a \in \mathbb{Z}_n^+$ плохим, если $\text{WITNESS}(a, n) = \text{“простое”}$.

Основная теорема

Теорема

Для нечетного составного числа n

$$|\{a \in \mathbb{Z}_n^+ : \text{WITNESS}(a, n) = \text{“составное”}\}| \geq \frac{n-1}{2}.$$

Идея доказательства

- Назовем основание $a \in \mathbb{Z}_n^+$ плохим, если $\text{WITNESS}(a, n) = \text{“простое”}$.
- Поскольку для плохого a выполнено $a^{n-1} \equiv 1 \pmod n$, все плохие элементы лежат в \mathbb{Z}_n^* .

Основная теорема

Теорема

Для нечетного составного числа n

$$|\{a \in \mathbb{Z}_n^+ : \text{WITNESS}(a, n) = \text{“составное”}\}| \geq \frac{n-1}{2}.$$

Идея доказательства

- Назовем основание $a \in \mathbb{Z}_n^+$ плохим, если $\text{WITNESS}(a, n) = \text{“простое”}$.
- Поскольку для плохого a выполнено $a^{n-1} \equiv 1 \pmod n$, все плохие элементы лежат в \mathbb{Z}_n^* .
- Мы хотим показать, что плохих оснований меньше половины.

Основная теорема

Теорема

Для нечетного составного числа n

$$|\{a \in \mathbb{Z}_n^+ : \text{WITNESS}(a, n) = \text{“составное”}\}| \geq \frac{n-1}{2}.$$

Идея доказательства

- Назовем основание $a \in \mathbb{Z}_n^+$ плохим, если $\text{WITNESS}(a, n) = \text{“простое”}$.
- Поскольку для плохого a выполнено $a^{n-1} \equiv 1 \pmod n$, все плохие элементы лежат в \mathbb{Z}_n^* .
- Мы хотим показать, что плохих оснований меньше половины.
- Для этого покажем, что все плохие элементы образуют собственную подгруппу \mathbb{Z}_n^* .

Простой случай

Простой случай

Простой случай

Простой случай

- Предположим, существует $x \in \mathbb{Z}_n^*$, такой что $x^{n-1} \not\equiv 1 \pmod n$.

Простой случай

Простой случай

- Предположим, существует $x \in \mathbb{Z}_n^*$, такой что $x^{n-1} \not\equiv 1 \pmod n$.
- Рассмотрим тогда $B = \{b \in \mathbb{Z}_n^* : b^{n-1} \equiv 1 \pmod n\}$.

Простой случай

Простой случай

- Предположим, существует $x \in \mathbb{Z}_n^*$, такой что $x^{n-1} \not\equiv 1 \pmod n$.
- Рассмотрим тогда $B = \{b \in \mathbb{Z}_n^* : b^{n-1} \equiv 1 \pmod n\}$.
- Ясно, что B является собственной подгруппой.

Простой случай

Простой случай

- Предположим, существует $x \in \mathbb{Z}_n^*$, такой что $x^{n-1} \not\equiv 1 \pmod n$.
- Рассмотрим тогда $B = \{b \in \mathbb{Z}_n^* : b^{n-1} \equiv 1 \pmod n\}$.
- Ясно, что B является собственной подгруппой.
- Таким образом, в дальнейшем предполагаем, что для всех $x \in \mathbb{Z}_n^*$

$$x^{n-1} \equiv 1 \pmod n.$$

Еще один простой случай

Еще один простой случай

Еще один простой случай

Еще один простой случай

- Предположим, $n = p^e$.

Еще один простой случай

Еще один простой случай

- Предположим, $n = p^e$.
- Тогда группа \mathbb{Z}_n^* является циклической, то есть существует элемент g , для которого $\text{ord}_n(g) = |\mathbb{Z}_n^*| = \varphi(n) = (p-1)p^{e-1}$.

Еще один простой случай

Еще один простой случай

- Предположим, $n = p^e$.
- Тогда группа \mathbb{Z}_n^* является циклической, то есть существует элемент g , для которого $\text{ord}_n(g) = |\mathbb{Z}_n^*| = \varphi(n) = (p-1)p^{e-1}$.
- Тогда из $g^{n-1} \equiv 1 \pmod n$ следует $n-1 \equiv 0 \pmod{\varphi(n)}$, но $n-1 \not\equiv 0 \pmod p$.

Еще один простой случай

Еще один простой случай

- Предположим, $n = p^e$.
- Тогда группа \mathbb{Z}_n^* является циклической, то есть существует элемент g , для которого $\text{ord}_n(g) = |\mathbb{Z}_n^*| = \varphi(n) = (p-1)p^{e-1}$.
- Тогда из $g^{n-1} \equiv 1 \pmod n$ следует $n-1 \equiv 0 \pmod{\varphi(n)}$, но $n-1 \not\equiv 0 \pmod p$.
- Итак, n представимо в виде произведения двух взаимно простых чисел $n_1, n_2 > 1$.

Основной случай

Основной случай

Основной случай

Основной случай

- Представим число $n - 1$ как $2^t u$, где u нечетно.

Основной случай

Основной случай

- Представим число $n - 1$ как $2^t u$, где u нечетно.
- Для каждого $a \in \mathbb{Z}_n^+$ рассмотрим последовательность вычетов по модулю n

$$\hat{a} = \langle a^u, a^{2u}, a^{2^2 u}, \dots, a^{2^t u} \rangle.$$

Основной случай

Основной случай

- Представим число $n - 1$ как $2^t u$, где u нечетно.
- Для каждого $a \in \mathbb{Z}_n^+$ рассмотрим последовательность вычетов по модулю n

$$\hat{a} = \langle a^u, a^{2u}, a^{2^2 u}, \dots, a^{2^t u} \rangle.$$

- Все эти элементы (кроме первого) важны тем, что они будут проверены процедурой WITNESS.

Основной случай

Основной случай

- Представим число $n - 1$ как $2^t u$, где u нечетно.
- Для каждого $a \in \mathbb{Z}_n^+$ рассмотрим последовательность вычетов по модулю n

$$\hat{a} = \langle a^u, a^{2u}, a^{2^2 u}, \dots, a^{2^t u} \rangle.$$

- Все эти элементы (кроме первого) важны тем, что они будут проверены процедурой WITNESS.
- Рассмотрим максимальное j , для которого существует такой элемент $v \in \mathbb{Z}_n^*$, что $v^{2^j u} \equiv -1 \pmod{n}$ (такое j существует, поскольку $(-1)^u \equiv -1 \pmod{n}$).

Основной случай

Основной случай

- Представим число $n - 1$ как $2^t u$, где u нечетно.
- Для каждого $a \in \mathbb{Z}_n^+$ рассмотрим последовательность вычетов по модулю n

$$\hat{a} = \langle a^u, a^{2u}, a^{2^2 u}, \dots, a^{2^t u} \rangle.$$

- Все эти элементы (кроме первого) важны тем, что они будут проверены процедурой WITNESS.
- Рассмотрим максимальное j , для которого существует такой элемент $v \in \mathbb{Z}_n^*$, что $v^{2^j u} \equiv -1 \pmod n$ (такое j существует, поскольку $(-1)^u \equiv -1 \pmod n$).
- Пусть $B = \{x \in \mathbb{Z}_n^* : x^{2^j u} \equiv \pm 1 \pmod n\}$.

Основной случай

Основной случай

- Представим число $n - 1$ как $2^t u$, где u нечетно.
- Для каждого $a \in \mathbb{Z}_n^+$ рассмотрим последовательность вычетов по модулю n

$$\hat{a} = \langle a^u, a^{2u}, a^{2^2 u}, \dots, a^{2^t u} \rangle.$$

- Все эти элементы (кроме первого) важны тем, что они будут проверены процедурой WITNESS.
- Рассмотрим максимальное j , для которого существует такой элемент $v \in \mathbb{Z}_n^*$, что $v^{2^j u} \equiv -1 \pmod n$ (такое j существует, поскольку $(-1)^u \equiv -1 \pmod n$).
- Пусть $B = \{x \in \mathbb{Z}_n^* : x^{2^j u} \equiv \pm 1 \pmod n\}$.
- Ясно, что это подгруппа. Покажем, что в ней лежат все плохие элементы.

Основной случай (продолжение)

Основной случай

Основной случай (продолжение)

Основной случай

- Рассмотрим последовательность \hat{a} плохого элемента a .

Основной случай (продолжение)

Основной случай

- Рассмотрим последовательность \hat{a} плохого элемента a .
- На последнем месте в ней стоит обязательно 1 . На предпоследнем — либо -1 , либо 1 (иначе был бы обнаружен нетривиальный корень из единицы).

Основной случай (продолжение)

Основной случай

- Рассмотрим последовательность \hat{a} плохого элемента a .
- На последнем месте в ней стоит обязательно 1 . На предпоследнем — либо -1 , либо 1 (иначе был бы обнаружен нетривиальный корень из единицы).
- Причем -1 не может стоять правее позиции j .

Основной случай (продолжение)

Основной случай

- Рассмотрим последовательность \hat{a} плохого элемента a .
- На последнем месте в ней стоит обязательно 1 . На предпоследнем — либо -1 , либо 1 (иначе был бы обнаружен нетривиальный корень из единицы).
- Причем -1 не может стоять правее позиции j .
- Значит, на j -м месте стоит 1 или -1 , то есть $a \in B$.

Основной случай (продолжение)

Основной случай

- Рассмотрим последовательность \hat{a} плохого элемента a .
- На последнем месте в ней стоит обязательно 1 . На предпоследнем — либо -1 , либо 1 (иначе был бы обнаружен нетривиальный корень из единицы).
- Причем -1 не может стоять правее позиции j .
- Значит, на j -м месте стоит 1 или -1 , то есть $a \in B$.
- Теперь осталось показать, что B — собственная подгруппа, то есть найти элемент \mathbb{Z}_n^* , не содержащийся в B .

Завершение доказательства

Завершение доказательства

Завершение доказательства

Завершение доказательства

- Возьмем элемент v , для которого $v^{2^j u} \equiv -1 \pmod{n}$.

Завершение доказательства

Завершение доказательства

- Возьмем элемент v , для которого $v^{2^j u} \equiv -1 \pmod{n}$.
- Тогда и $v^{2^j u} \equiv -1 \pmod{n_1}$ (поскольку $n = n_1 n_2$ и $(n_1, n_2) = 1$).

Завершение доказательства

Завершение доказательства

- Возьмем элемент v , для которого $v^{2^j u} \equiv -1 \pmod{n}$.
- Тогда и $v^{2^j u} \equiv -1 \pmod{n_1}$ (поскольку $n = n_1 n_2$ и $(n_1, n_2) = 1$).
- Существует элемент w , для которого $w \equiv v \pmod{n_1}$ и $w \equiv 1 \pmod{n_2}$.

Завершение доказательства

Завершение доказательства

- Возьмем элемент v , для которого $v^{2^j u} \equiv -1 \pmod{n}$.
- Тогда и $v^{2^j u} \equiv -1 \pmod{n_1}$ (поскольку $n = n_1 n_2$ и $(n_1, n_2) = 1$).
- Существует элемент w , для которого $w \equiv v \pmod{n_1}$ и $w \equiv 1 \pmod{n_2}$.
- Тогда $w^{2^j u} \equiv -1 \pmod{n_1}$ и $w^{2^j u} \equiv 1 \pmod{n_2}$.

Завершение доказательства

Завершение доказательства

- Возьмем элемент v , для которого $v^{2^j u} \equiv -1 \pmod{n}$.
- Тогда и $v^{2^j u} \equiv -1 \pmod{n_1}$ (поскольку $n = n_1 n_2$ и $(n_1, n_2) = 1$).
- Существует элемент w , для которого $w \equiv v \pmod{n_1}$ и $w \equiv 1 \pmod{n_2}$.
- Тогда $w^{2^j u} \equiv -1 \pmod{n_1}$ и $w^{2^j u} \equiv 1 \pmod{n_2}$.
- Но тогда $w^{2^j u} \not\equiv \pm 1 \pmod{n}$.

Завершение доказательства

Завершение доказательства

- Возьмем элемент v , для которого $v^{2^j u} \equiv -1 \pmod{n}$.
- Тогда и $v^{2^j u} \equiv -1 \pmod{n_1}$ (поскольку $n = n_1 n_2$ и $(n_1, n_2) = 1$).
- Существует элемент w , для которого $w \equiv v \pmod{n_1}$ и $w \equiv 1 \pmod{n_2}$.
- Тогда $w^{2^j u} \equiv -1 \pmod{n_1}$ и $w^{2^j u} \equiv 1 \pmod{n_2}$.
- Но тогда $w^{2^j u} \not\equiv \pm 1 \pmod{n}$.
- Поскольку $(w, n_1) = 1$ и $(w, n_2) = 1$, то $(w, n) = 1$. Следовательно, $w \in \mathbb{Z}_n^*$, но $w \notin B$.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

минимальный разрез выбирать случайное ребро и стягивать его в вершину

Что мы узнали за сегодня?

Что мы узнали за сегодня?

минимальный разрез выбирать случайное ребро и стягивать его в вершину

проверка простоты числа выбирать случайный вычет и проверять, не является ли он свидетелем того, что число составное

Спасибо за внимание!