АМВ. ДЗ на неделю 11.

Прохоров Юрий, 771

Задача 1

В протоколе RSA выбраны $p = 17$, $q = 23$, $N = 391$, $e = 3$. Выберите ключ d и зашифруйте сообщение 41. Затем расшифруйте полученное сообщение и убедитесь, что получится исходное 41.

Решение:

Протокол RSA:

- $N = pq,\ p, q$ — большие простые числа
- $(N, e) = (391, 3)$ — открытый ключ, e выбирает получатель:

$$\gcd(e, (p-1)(q-1)) = 1$$

- $d = e^{-1} \mod (p-1)(q-1)$ — секретный ключ
- Посылаемое сообщение x кодируется:

$$y = x^e \mod N$$

- Принимаемое сообщение y декодируется:

$$x = y^d \mod N$$

С помощью алгоритма Евклида найдем секретный ключ d:

$$d = 3^{-1} \bmod 16 \cdot 22 = 3^{-1} \bmod 352$$

$$3d \equiv 1 \bmod 352$$

$$3d + 352k = 1$$

$$d = -117 + 352t, \qquad k = 1 - 3t \qquad \Longrightarrow \qquad d = 235$$

Зашифруем сообщение $x = 41$ (вычисления по модулю $N = 391$):

$$y = 41^3 = 105$$

Дешифруем сообщение:

$$x = 105^{235} = 41$$

Вычисление можно провести, используя бинарное возведение в степень.

Задача 2

Пусть в протоколе RSA открытый ключ (N, e), $e = 3$. Покажите, что если злоумышленник узнаёт закрытый ключ d, то он может легко найти разложение N на множители.

Решение:

Source of inspiration.

Лемма 1. *Пусть $N = pq,\ p, q$ — нечетные простые числа. Тогда уравнение*

$$x^2 \equiv 1 \bmod N$$

имеет 4 (необязательно различных) решения: $1, -1, a, -a$*, где число* a *— единственное решение*

$$\begin{cases} a \equiv 1 \bmod q \\ a \equiv -1 \bmod p \end{cases}$$

$$a = \left[p\left(p^{-1} \bmod q\right) - q\left(q^{-1} \bmod p\right) \right] \bmod N$$

Доказательство:

Запишем исходное сравнение в виде:

$$(x-1)(x+1) \equiv 0 \bmod N$$

По китайской теореме об остатках:

$$\mathbb{Z}_N \cong \mathbb{Z}_p \otimes \mathbb{Z}_q \qquad \Longrightarrow \qquad (x-1)(x+1) = 0 \longleftrightarrow (0,0) \qquad \Longrightarrow$$

$$\begin{cases} (x-1)(x+1) \equiv 0 \bmod q \\ (x-1)(x+1) \equiv 0 \bmod p \end{cases}$$

Так как p, q — простые, то кольца $\mathbb{Z}_p$ и $\mathbb{Z}_q$ являются полями, а в них нет делителей нуля. Поэтому

$$\begin{cases} x-1 \equiv 0 \bmod q \qquad \text{или} \qquad x+1 \equiv 0 \bmod q \\ x-1 \equiv 0 \bmod p \qquad \text{или} \qquad x+1 \equiv 0 \bmod p \end{cases}$$

Имеем 4 случая. По китайской теореме об остатках, каждый из них имеет единственное решение. Легко видеть, что $x = 1$ — решение, если взять два левых уравнения, а $x = -1 = N - 1$ есть решение, если взять два правых. Рассмотрим нетривиальный случай

$$\begin{cases} x \equiv 1 \bmod q \\ x \equiv -1 \bmod p \end{cases}$$

По формуле из китайской теоремы об остатках, решением является число

$$a = \left[1 \cdot p\left(p^{-1} \bmod q\right) + (-1) \cdot q\left(q^{-1} \bmod p\right) \right] \bmod N$$

По этой формуле видно, что решением второго нетривиального случая будет $-a$. □

Лемма 2. *Пусть* $N = pq,\ p, q$ *— нечетные простые числа, и число* x $(x \neq 1,\ x \neq -1)$ *таково, что*

$$x^2 \equiv 1 \bmod N.$$

Тогда число $b = \gcd\left(x+1, N\right)$ *равно либо* p*, либо* q*.*

Доказательство:

По лемме 1, число x есть, для определенности (при необходимости можно вместо x рассмотреть $-x$), решение системы сравнений

$$\begin{cases} x \equiv 1 \bmod q \\ x \equiv -1 \bmod p \end{cases}$$

Тогда $x + 1 = \alpha p$. Но $0 \leq x < pq$, поэтому $0 \leq \alpha \leq q$. Но

$$\alpha = q \qquad \Longrightarrow \qquad x = N - 1 = -1$$

$$\alpha = 0 \qquad \Longrightarrow \qquad x = -1$$

Поэтому $0 < \alpha < q$. Но тогда $\gcd\left(x+1, N\right) = \gcd\left(\alpha p,\ qp\right) = p$. □

В задаче нам известны числа e, d, N. Из леммы 2 следует, что если мы сможем найти нетривиальный корень из единицы по модулю N, то мы сможем найти множители p и q. Перебирать все числа очень долго, поэтому нужно использовать данные нам числа e и d. Мы знаем, что

$$x^{ed} \equiv x \bmod N \qquad \Longleftrightarrow \qquad x^{ed-1} \equiv 1 \bmod N \qquad \forall x$$

Попробуем извлечь корень из числа x^{ed-1}. Числа e, d — взаимно простые с числом $(p-1)(q-1)$, которое делится на 4. Поэтому e и d — нечетные числа, значит число $ed-1$ является четным. Значит, мы можем легко извлечь корень из x^{ed-1}.

Иногда нам может повезти, и можно несколько раз извлечь корень. Так как $ed-1$ четно, то пусть

$$ed - 1 = 2^t r, \qquad t > 0,\ r - \text{нечетное}$$

Посчитаем массив чисел

$$A:\ x^r,\ x^{2r},\ x^{2^2 r},\ \ldots,\ x^{2^t r}, \qquad A_j = x^{2^j r}$$

Последнее всегда равно 1. Если в массиве A есть такая пара *последовательных* чисел z и z^2, что

$$z \neq \pm 1, \qquad z^2 = 1,$$

то мы нашли нетривиальный корень из единицы и можем посчитать p, q.

Остается вопрос, как выбрать такое число x. Будем выбирать его случайно: $1 < x < N$. Найдем вероятность, что в массиве A есть нетривиальный корень из 1.

$A_t = 1$. По лемме 1, существует 4 корня из единицы. Можно оценить, что с вероятностью $\frac{1}{4}$ $A_{t-1} = -1$ (тогда надо генерировать x заново), с вероятностью $\frac{1}{4}$ $A_{t-1} = 1$ (тогда переходим к следующему элементу) и с вероятностью $\frac{1}{2}$ $A_{t-1} \neq \pm 1$ (тогда наши разложение). То есть с вероятностью, не хуже некоторой фиксированной массив A содержит нетривиальный корень из единицы. Значит, с большой за конечное число шагов мы найдем разложение.

Алгоритм:

1. $ed - 1 = 2^t r$.
2. Сгенерировать случайное число $1 < x < N$.
3. Вычислять числа
$$A_0 = x^r,\ A_1 = A_0^2, \ldots,\ A_i = A_{i-1}^2, \qquad i = 1, \ldots, t$$
пока не найдется нетривиальный корень из единицы.
4. Если такой корень a нашелся, то
$$p = \gcd\big(a + 1, N\big), \qquad q = \frac{N}{p}.$$
5. Если такой корень не нашелся, то перейти к пункту 2.

Задача 3

Схема RSA позволяет также создавать защищенные электронные подписи. Если открытый ключ (N, e), то автор сообщения, обладающий закрытым ключом d, отправляет сообщение A^d, где A — незашифрованное сообщение. После этого идентификация подписи - это возведение в степень e. Пусть открытый ключ $(2021, 25)$. В какую степень автору нужно возвести сообщение, чтобы отправить его за своей электронной подписью?

Решение:

Автор должен возвести его в степень d.

$$N = 2021 = 43 \cdot 47 \qquad \Longrightarrow \qquad d = e^{-1} \bmod (p-1)(q-1) = 25^{-1} \bmod 1932 = 541$$

Задача 4

В памяти хранится массив чисел $A[1,\ldots,n]$. Назовем **горкой** элемент $A[i]$, который не меньше обоих своих соседей, если $1 < i < n$, или не меньше своего правого или левого соседа, если $i = 1$ или $i = n$.

(i) Постройте как можно более быстрый алгоритм, использующий попарные сравнения, находящий "горку" в A, докажите его корректность и оцените число сравнений.

(ii) Приведите как можно более точную $\Omega(\cdot)$–оценку числа попарных сравнений, которые должен использовать любой алгоритм, находящий "горку" посредством попарных сравнений.

Чтобы получить полный балл за эту задачу, время работы алгоритма из первого пункта должно соответствовать теоретической нижней оценке, которую нужно получить во втором пункте.

Решение:

Заметим, что максимум массива всегда является горкой. Для нахождения максимума нужно необходимо и достаточно $n-1$ сравнений, поэтому $n-1$ сравнений хватит, чтобы найти горку. Однако, чтобы найти горку, необязательно искать максимум.

(i) Алгоритм:

1. Если $n = 1$, то a_1 — горка.
2. Если $n = 2$, то горка находится за 1 сравнение.
3. Если $n > 2$:
   - Сравнить $a_{\lfloor \frac{n}{2} \rfloor}$ с левым соседом.
   - Если левый сосед больше или равен, то рекурсивно найти горку в подмассиве $A\left[1,\ldots,\lfloor \frac{n}{2} \rfloor - 1\right]$.
   - Если левый сосед меньше, то сравнить с правым соседом.
   - Если правый сосед больше или равен, то рекурсивно найти горку в подмассиве $A\left[\lfloor \frac{n}{2} \rfloor + 1,\ldots,n\right]$.
   - Если правый сосед меньше, то $a_{\lfloor \frac{n}{2} \rfloor}$ — горка.

Оценим число сравнений. В худшем случае производится рекурсивный вызов от правого подмассива, в котором, если n четно, ровно в два раза меньше элементов. Если n нечетно, то там $\frac{n-1}{2}$ элементов. Наиболее плохая оценка достигается, если $n = 2^k$.

$$T(n) = T\left(\frac{n}{2}\right) + 2,\ n = 2^k \qquad \Longrightarrow \qquad f(k) = f(k-1) + 2,\ k > 2, \qquad f(1) = 1$$

Частное решение рекурренты:

$$f_0(k) = 2k$$

Решение однородной рекурренты: $f(k) = f(k-1)$:

$$f_{\text{одн.}} = c \qquad \Longrightarrow \qquad f(k) = 2k + c,\ f(1) = 2 + c = 1$$

$$f(k) = 2k - 1$$

Итак, для любого $n > 1$ алгоритм делает не более

$$T(n) = 2\log_2 n - 1 = \Theta(\log n)$$

сравнений.

Докажем корректность. Докажем следующее утверждение: *если левый сосед $\geq$ центрального элемента, то в левом подмассиве есть горка.*

Пусть a_{k+1} — центральный элемент, $a_k \geq a_{k+1}$. Требуется показать, что среди $a_1,\ldots,a_k$ есть горка. Допустим, в подмассиве горки нет. Тогда a_k — максимум подмассива, значит, $a_{k-1} \leq a_k$. Но $a_k \geq a_{k+1}$, поэтому a_k — горка.

Аналогично доказывается, что есть центральный элемент $\leq$ правого соседа, то в правом подмассиве есть

горка. Если оба сравнения показали, что центральный элемент больше, то тогда он является горкой, что алгоритм и выводит.

(*ii*) Покажем, что любому алгоритму нужно $\Omega(\log n)$ сравнений, чтобы найти какую-нибудь горку. Другими словами, для любого n существует вход, на котором алгоритм будет делать $\geq C \log n$ сравнений.

Алгоритм возвращает номер элемента-горки. Возможны случаи, когда все ответы правильные (все элементы равны) и случаи, когда правильный ответ только один. Рассмотрим подмножество входов, для который правильный ответ единственен. Найдем нижнюю оценку только для этих входов. Нижняя оценка в общем случае будет не меньше.

Рассмотрим разрешающее дерево алгоритма. В его листьях: элементы $a_1, \ldots, a_n$, — ответ, который выдает алгоритм. Высота этого дерева: $\log_2 n = \Theta(\log n)$. Поэтому, чтобы узнать единственный правильный ответ из n вариантов, алгоритму понадобится $\Omega(\log n)$ сравнений.

Задача 5

Пусть $G(V, E)$ — простой неориентированный граф, множество вершин которого допускает дизъюнктное разбиение на непересекающиеся подмножества $V = S \sqcup T$, такие, что индуцированные подграфы G_S и G_T являются кликами.

Верно ли, что соответствующий язык всех графов, обладающих таким свойством, принадлежит $\mathcal{NP}$-*complete*?

По определению, **индуцированный подграф** G_{V_1}, $V_1 \subseteq V(G)$ *имеет вершинами множество* V_1*, а ребрами — все ребра* G *с вершинами из* V_1*.*

Решение:

Пусть L — данный язык. Покажем, что $L \in \mathcal{P}$, сведя задачу к языку двудольных графов (или, эквивалентно, 2-$COLOUR$).

Докажем, что граф $G \in L$ тогда и только тогда, когда его дополнение $\overline{G}$ является двудольным графом.

Пусть $G \in L$, G_S, G_T — клики, $V = S \sqcup T$. Тогда $\overline{G_S}, \overline{G_T}$ — антиклики (независимые множества), причем $V = S \sqcup T$. Значит, эти вершины являются долями двудольного графа $\overline{G}$.

Пусть $\overline{G}$ — двудольный граф. Тогда $\overline{G_S}, \overline{G_T}$ — антиклики $\Longrightarrow$ G_S, G_T — две клики, вершины которые образуют дизъюнктное разбиение V. Значит, $G \in L$.

Таким, образом, полиномиальная сводимость

$$L \leq_p 2\text{-}COLOUR = BIPARTITE \in \mathcal{P}$$

заключается в построении $\overline{G}$ по G.

Задача 6

Пусть $L = \{(\langle G \rangle, s, t)\}$ — это язык, состоящий из стандартных описаний неориентированных графов G, в которых выделены различные вершины s и t такие, что для любого $S \geq 10$ существует путь из s в t длины S. Проверить принадлежность языка классу *co*-$\mathcal{NP}$.

Длина пути равна числу рёбер в нем, а в пути допускается повторение вершин и повторение ребер, то есть можно, например, возвращаться по ребру, по которому только что был сделан переход.

Решение:

Утв. 1. *Пусть в* G *существует путь* $s \to t$ *длины* n*. Тогда для* $\forall k \geq 0$ *существует путь* $s \to t$ *длины* $n + 2k$*.*

Доказательство:
Достаточно взять первое ребро пути и сходить по нему k раз туда-обратно. Затем пройти по оставшейся части пути. □

Утв. 2. *В G существует путь $s \to t$ длины S для $\forall S \geq n_0$ тогда и только тогда, когда существуют пути $s \to t$ длины n_0 и $n_0 + 1$.*

Доказательство:
Необходимость очевидна. Достаточность следует из утверждения 1. □

Утв. 3. *В графе G можно за полиномиальное время найти кратчайшие пути четной и нечетной длины.*

Доказательство:
Построим по графу $G = (V, E)$ граф $G' = (V', E')$:

- $\forall v \in V: \ v_0, v_1 \in V'$
- $\forall [u, v] \in E: \ [u_0, v_1], \ [u_1, v_0] \in E'$

Найдем с помощью BFS за $O(|V| + |E|)$ в G' кратчайшие пути из s_0 до всех вершин. Пусть в G был некоторый путь четной длины $s \to t$. Тогда ему соответствует путь той же четной длины в $G': \ s_0 \to t_0$. Пути нечетной длины $s \to r$ в G соответствует путь той же нечетной длины в $G': \ s_0 \to t_1$.

Таким образом, если мы найдем кратчайший путь $s_0 \to u_0$ в G', то он будет иметь четную длину и соответствовать кратчайшему четному пути $s \to u$ в G. Кратчайший путь $s_0 \to u_1$ в G' соответствует кратчайшему нечетному пути $s \to u$ в G. □

Из трех утверждений выше соберем полиномиальный алгоритм проверки принадлежности $\big(G, s, t\big) \in L$. Найдем кратчайшие пути в G четной и нечетной длины из s в t. Пусть они имеют длины l_0 и l_1 соответственно. Из утверждения 2 следует, что

$$\begin{cases} l_0 \leq 10 \\ l_1 \leq 11 \end{cases} \qquad \Longleftrightarrow \qquad \big(G, s, t\big) \in L$$

Алгоритм:

1. Найти длины l_0 и l_1 кратчайших путей $s \to t$ в G четной и нечетной длин соответственно с помощью алгоритма из утверждения 3.
2. Если $l_0 \leq 10 \ \wedge \ l_1 \leq 11$, то $\big(G, s, t\big) \in L$, иначе $\big(G, s, t\big) \notin L$.

Мы показали, что $L \in \mathcal{P} \subseteq$ *co-*$\mathcal{NP}$.

Задача 7

Дан неориентированный граф G без петель и кратных рёбер, имеющий m рёбер, которым приписаны положительные веса.

Раскрасим вершины в два цвета, **трудностью раскраски** назовем наибольший вес ребра между вершинами одного и того же цвета, а если таких рёбер нет, то трудность раскраски считаем равной нулю.

Постройте и обоснуйте $O(m \log m)$–алгоритм, находящий раскраску с **наименьшей трудностью**.

Решение:

Построение 2-раскраски эквивалентно дизъюнктному разбиению вершин V графа на два множества. При этом трудностью раскраски будет являться вес максимального ребра в одной из долей.

Будем считать, что мы умеем за $O(m)$ проверять, является ли граф двудольным (можно пытаться жадно красить вершины в два цвета).

Докажем, *что наименьшая трудность раскраски равна w тогда и только тогда, когда граф G с удаленными из него ребрами веса $\leq w$ (граф G') является двудольным (w — наименьшее возможное).*

Пусть w — наименьшая трудность раскраски. Раскрасим граф G этим способом. Удалим все ребра веса $\leq w$. Так как w — максимальный вес ребра между вершинами одного цвета, то в новом графе G' нет ребер между вершинами одного цвета. Значит, он является двудольным.

Пусть w — такое наименьшее число, что после удаления ребер веса $\leq w$ граф G стал двудольным. Найдем 2-раскраску графа G'. Теперь вернем удаленные ребра, оставив ту же раскраску. Трудность этой раскраски равна w, т.к. w — максимальный вес удаленного ребра. Если бы существовала раскраска с меньшей трудностью, то и w можно бы было выбрать меньше.

Итак, задача сводится к нахождению минимального веса w, что если удалить все ребра веса $\leq w$, то граф станет двудольным.

Алгоритм:

1. Отсортировать ребра.
2. С помощью бинарного поиска найти минимально возможное w. На каждой итерации бинарного поиска:
   - Удалить из графа все ребра веса $\leq w$.
   - Проверить, является ли граф двудольным.

Корректность алгоритма обоснована выше. Оценим время. На сортировку требуется $O(m \log m)$ операций. Бинарный поиск в отсортированном массиве из m элементов требует $\log m$ итераций. На каждой из них нужно время $O(m)$ на удаление ребер и $O(m)$ для проверки на двудольность. Итого:

$$T(m) = O(m \log m) + \log m \Big(O(m) + O(m) \Big) = O(m \log m).$$