Дисклеймер.

Автор не несет ответственности за любой ущерб, причиненный Вам при использовании данного документа. Автор напоминает, что данный документ может содержать ошибки и опечатки, недостоверную и/или непроверенную информацию. Если Вы желаете помочь в развитии проекта или сообщить об ошибке/опечатке/неточности:

GitHub проекта

Автор в ВК

Содержание

1 Некоторые определения из теории множеств. Прямое произведение, разбиение множеств. Мощность объединения

Определение. Мощность множества — число элементов, входящих в него. Обозначается $|A|$.

Определение. Прямое произведение — множество всех возможных упорядоченных пар (i, j), где $i \in A,\ j \in B$. Обозначается как $A \times B$. Мощность этого множества равна произведению мощностей: $|A \times B| = |A| \cdot |B|$.

Определение. Совокупность $A_1, ..., A_n$ непустых попарно дизъюнктных подмножеств множества M называется его разбиением, если M равно их объединению.

Определение. Разбиение $\mathscr{A} = \{A_1, ...A_n\}$ множества M называется измельчением разбиения $\mathscr{B} = \{B_1, ..., B_k\}$, если каждое из множеств A_iполностью содержится в одном из множеств B_j, то есть $\forall i \in 1 : n\ \exists! j \in 1 : k$, такое, что $A_i \subset B_j$. Тогда будем говорить, что $\mathscr{B}$ крупнее $\mathscr{A}$ и записывать $\mathscr{A} \sqsubset \mathscr{B}$.

Определение. $\mathscr{A} = \{A_1, ..., A_k\}$ и $\mathscr{B} = \{B_1, ..., B_l\}$ — разбиения множества M. Произведение разбиений $\mathscr{A}$ и $\mathscr{B}$ называется разбиение $\mathscr{C} = \{C_1, ...C_m\}$, если оно является измельчением этих разбиений, причем самым крупным из таких разбиений.

Теорема. *Произведение двух разбиений существует.*

Доказательство. Предъявим множество, являющееся произведением произвольных $\mathscr{A}, \mathscr{B}$. Возьмем все множества $D_{ij} = A_i \cap B_j,\ i \in 1 : k,\ j \in 1 : l$. Подмножества D_{ij} дизъюнктны и их объединение составляет множество M. Отбросив пустые множества получим разбиение $\mathscr{C}$.

Докажем, что оно самое крупное. Пусть имеется разбиение $\mathscr{F} = \{F_1, ..., F_t\}$, которое мельче, чем $\mathscr{A}, \mathscr{B}$. Покажем,что оно мельче, чем $\mathscr{C}$. $\forall F_k\ \exists A_{i_k},\ B_{j_k}$, такие, что $F_k \subset A_{i_k},\ F_k \subset B_{j_k} \Rightarrow F_k \subset A_{i_k} \cap B_{j_k} \Rightarrow F_k \subset C_k$, следовательно, $\mathscr{F}$ мельче $\mathscr{C}$. □

Теорема. *Формула включений-исключений*

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_i |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_i \cap A_k| - ... + (-1)^{n-1} |A_1 \cap ... \cap A_n|$$

Доказательство. Индукция по n. При $n = 1$ очевидно. Пусть верно для n, рассмотрим для $n + 1$:

$$\left|\bigcup_{i=1}^{n+1} A_i\right| = |A_1 \cup ... \cup A_n| + |A_{n+1}| - \left|\bigcup_{i=1}^{n} (A_i \cap A_{n+1})\right| =$$

$$= \sum_i^n |A_i| - \sum_{i<j}^n |A_i \cap A_j| + \sum_{i<j<k}^n |A_i \cap A_i \cap A_k| - ... + (-1)^{n-1} |A_1 \cap ... \cap A_n| +$$

$$+ |A_{n+1}| - \left|\bigcup_{i=1}^{n} (A_i \cap A_{n+1})\right| = \sum_i^{n+1} |A_i| - \sum_{i<j}^n |A_i \cap A_j| + \sum_{i<j<k}^n |A_i \cap A_i \cap A_k| - ... + (-1)^{n-1} |A_1 \cap ... \cap A_n| +$$

$$+ \sum_i^n |A_i \cap A_{n+1}| - \sum_{i<j}^n |A_i \cap A_j \cap A_{n+1}| + \sum_{i<j<k}^n |A_i \cap A_i \cap A_k \cap A_{n+1}| - ... + (-1)^n |A_1 \cap ... \cap A_{n+1}| =$$

$$\sum_i^{n+1} |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_i \cap A_k| - ... + (-1)^n |A_1 \cap ... \cap A_{n+1}|$$

□

2 Алгоритм перебора перестановок. Нумерация перестановок

Определение. Перестановкой из k элементов называется упорядоченный набор из k различных чисел диапазона $1 : k$.

Обозначим через T_k прямое произведение множеств вида $M_1 \times ... \times M_k$, где каждое M_i состоит из целых чисел от 1 до $i-1$. Теперь обозначим множество перестановок из k элементов как P_k и построим взаимно однозначное соответствие между P_k и T_k: возьмем перестановку $(r_1, ..., r_k)$ и найдем число значений, меньших r_i среди $r_{i+1}, ..., r_k$. Это число примем за t_i. Таким образом:

r_i	4	8	1	5	7	2	3	6
t_i	3	6	0	2	3	0	0	0

Если использовать это отображение при переборе, то перестановки будут перебираться в лексикографическом порядке, то есть перестановка $(r_1, ..., r_k)$ предшествует перестановке $(R_1, ..., R_k)$, если до какого-то индекса d элементы этих перестановок совпадают, а далее $r_d < R_d$.

Из того, что перестановки следуют в лексикографическом порядке, получаем простой способ получения следующей перестановки из данной:

1) В заданной перестановке $(r_1, ..., r_k)$ найти ее наибольший суффикс (участок «хвоста» перестановки), в котором элементы расположены по убыванию (максимальность суффикса означает, что $r_{t-1} < r_t$).

2) Выбрать в $(r_t, ..., r_k)$ элемент, следующий по величине за r_{t-1} и поставить его на место $t-1$. Оставшиеся элементы, включая r_{t-1} расположить за ним в порядке возрастания.

Однако таким образом перестановки меняются очень сильно. Мы хотим разработать алгоритм, где следующая перестановка получалась из предыдущей транспозицией двух соседних элементов. Будем передвигать k-ый элемент на соседнее место. Для того, чтобы понять, передвигать его вправо или влево, используем механизм, который на каждые $k-1$ итерации будет давать команду на сдвиг k-го элемента, а затем менять направление его движения на противоположный и давать команду на сдвиг элемента с меньшим номером.

Объясним алгоритм на русском языке, а не на этой помеси брэйнфака и уайтспейса. Выбираем последний элемент перестановки и транспозициями «гоним» его на первое место. Затем меняем местами последний и предпоследний элемент перестановки и «гоним» наш (теперь уже первый) элемент назад, до последнего места. Меняем местами первый и второй элементы перестановки, повторяем, пока не получим все перестановки. Для маленькой перестановки (1 2 3):

(1 2 3) → (1 3 2) → (3 1 2) → (3 2 1) → (2 3 1) → (2 1 3).

Оба этих алгоритма являются алгоритмами перебора перестановок.

Теперь поговорим о нумерации перестановок. Здесь нам понадобится то самое множество T_k. Итак, алгоритм получения номера перестановки прост, как яйца пингвина: берем перестановку, составляем для нее кортеж $(t_1, ..., t_k)$, затем получаем номер по формуле

$$\sum_{i=1}^{n} t_i \cdot (n-i)$$

на примере: $P_3 = (2\ 3\ 1)$, для него $T_3 = (1\ 1\ 0)$, по формуле: $N = 1 \cdot 2 + 1 \cdot 1 + 0 \cdot 1 = 3$, что верно, если перебирать данные перестановки в лексикографическом порядке.

3 Числа Фибоначчи. Теорема о представлении

Определение. Числа Фибоначчи — последовательность чисел, определяемая следующим условием: $F_0 = 0;\ F_1 = 1;\ F_n = F_{n-1} + F_{n-2}$.

Таким образом получаем $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...$

Числа Фибоначчи имеют много замечательных свойств, к примеру, приближают отношение золотого сечения. Положим $\varphi_n = {}^{F_{n+1}}/_{F_n}$, перейдем к пределу в отношении ${}^{F_{n+1}}/_{F_n} = {}^{F_n}/_{F_n} + {}^{F_{n-1}}/_{F_n}$ и получим $\varphi = 1 + \varphi^{-1}$, откуда $\varphi = \frac{\sqrt{5}+1}{2} \approx 1.61834...$

Теорема. *Любое натуральное число s однозначно представляется в виде суммы чисел Фибоначчи $s = F_{i_0} + ... + F_{i_r}$, где $i_0 = 0$ и $\forall k \in 1 : r$ верно $i_{k-1} + 1 < i_k$.*

Доказательство. Докажем, что представление существует. Пусть $j(s)$ — номер максимального числа Фибоначчи, не превосходящего s. Положим $s' = s - F_{j(s)}$. Из определения $j(s)$ следует, что $s' < F_{j(s)-1}$. Теперь получим искомое представление s как сумму $F_{j(s)} + s'$, где s' имеет свое представление.

Докажем однозначность представления. Пусть имеется еще одно представление $s = F_{j_0} + .. + F_{j_q}$. НУО можно считать, что $j_q < j(s)$, так как больше быть не может, а рассматривать равные не имеет смысла в текущем контексте. Если заменить F_{j_q} на $F_{j(s)-1}$, сумма разве что увеличится. Аналогично заменим предпоследнее слагаемое на $F_{j(s)-3}$, но в любом случае сумма не сможет подняться до значения $F_{j(q)}$, следовательно, другое представление невозможно. □

Запись числа в фибоначчиевой системе счисления осуществляется так: раскладываем число в сумму чисел Фибоначчи, записываем все числа Фибоначчи до наибольшего из суммы, затем наличие определенного числа Фибоначчи в сумме обозначается 1, а отсутствие 0. К примеру, $51 = 34 + 13 + 3 + 1$, тогда

0	1	1	2	3	5	8	13	21	34
1	0	1	0	1	0	0	1	0	1

4 Условные вероятности и формула Байеса

Определение. Отношение благоприятных равновозможных исходов к числу всех исходов называется вероятностью какого-либо события.

Предложение. *Разбиение множества элементарных событий S на несовместимые события $S_1, ..., S_n$ называется полной системой событий. Каждое событие A может быть представлено в как объединение несовместимых событий. Очевидно равенство:*

$$P(A) = \sum_{i \in 1:n} P(A \cap S_i)$$

которое называется формулой полной вероятности.

Определение. События называются независимыми, если вероятность их совмещения равна произведению их вероятностей.

Предложение. *Пусть A — некоторое событие, имеющее положительную вероятность. Определим для каждого события B условную вероятность B при условии A как*

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

такого что $P(B \cap A) = P(B|A) \cdot P(A)$.

Определение. Условную вероятность независимых событий можно определить как $P(B|A) = P(B)$.

Предложение. *Используя условные вероятности можно видоизменить формулу полных вероятностей, заменив в ней $P(B \cap A_i)$ на $P(B|A_i)P(A_i)$:*

$$P(B) = \sum_i P(B|A_i)P(A_i)$$

Предложение. *Также формулу полных вероятностей можно изменить следующим образом: выразив $P(B \cap A_i)$ с помощью условных вероятностей, получим $P(A_i|B)P(B) = P(B|A_i)P(A_i)$. Разделив обе части формулы на $P(B)$ и расписав его как $P(B) = \sum P(B|A_i)P(A_i)$, получим формулу Байеса:*

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum(P(B|A_j)P(A_j))}$$

формула Байеса используется для вычисления так называемых апостериорных вероятностей, то есть позволяет по известному факту произошедшего события сказать, с какой вероятностью оно было вызвано данной причиной.

5 Математическое ожидание и дисперсия случайной величины

Определение. Пусть $\Omega = \{\omega\}$ — множество элементарных событий, p_ω — вероятность элементарного события ω. Заданная на Ω функция $f : \Omega \to R^1$ называется случайной величиной.

Определение. Для задания случайной величины важен набор вероятностей, с которыми случайная величина принимает те или иные значения. Этот набор вероятностей называется распределением случайной величины.

Определение. Случайные величины называются независимыми, если независимы события, которые их вызывают: $P(\xi = a, \mu = b) = P(\xi = a) \times P(\mu = b)$.

Определение. Величина

$$E\xi = \sum_\omega \xi(\omega)p_\omega = \sum_a a \times P(\xi = a)$$

называется математическим ожиданием случайной величины ξ. Ближайшая аналогия — центр тяжести для массы точек.

Отсюда следуют свойства математического ожидания:
1) $Ec = c$, где $c = const$.
2) Если $\mu = b\xi$, где b — константа, то $E\mu = bE\xi$.
3) $\xi : a_i p_i,\ \mu : b_i, q_i$ — независимы, тогда:
$E(\xi \cdot \mu) = \sum_i \sum_j a_i b_j P(\xi = a_i, \mu = b_j) = \sum a_i P(\xi = a_i) \sum b_j P(\mu = b_j) = E(\xi) + E(\mu)$.
4) $E(\xi, \mu) = \sum_i \sum_j (a_i + b_j)P(\xi = a_i, \mu = b_j) = \sum_i \sum_j a_i P(\xi = a_i, \mu = b_j) + \sum \sum b_j P(\xi = a_i, \mu = b)_j = \sum \sum a_i P(\xi = a_i|\mu = b_j)P(\xi = a_i) + \sum \sum b_j P(\mu = b_j|\xi = a_i)P(\mu = b_j) = \sum_i a_i P(\xi = a_i) \cdot \sum_i P(\xi = a_i|\mu = b_j) + \sum_j b_j P(\nu = b_j) \cdot \sum P(\mu = b_j|\xi = a_i) = E\xi + E\mu$.

Определение. Математическое ожидание квадрата отклонения случайной величины от математического ожидания называется дисперсией случайной величины:

$$D\xi = E(\xi - E\xi)^2$$

Предложение. $D\xi = E(\xi^2 - 2\xi E\xi + (E\xi)^2) = E\xi^2 - 2E(\xi E\xi) + E((E\xi)^2) = E\xi^2 - 2(E\xi)^2 + (E\xi)^2$ *(из того, что $E(E\xi) = E\xi$).*

Свойства дисперсии:
1) $Dc = 0$
2) $D(c\xi) = E(c\xi)^2 - (E(c\xi))^2 = c^2E\xi^2 - c^2(E\xi)^2 = cD\xi$.
3) ξ, μ — независимы. Тогда $D(\xi + \mu) = E(\xi + \mu)^2 - (E(\xi + \mu))^2 = E(\xi^2 + 2\xi\mu + \mu^2) - (E(\xi) + E(\mu)^2 = E\xi^2 + 2E(\xi\mu) + E\mu^2 - (E\xi)^2 - 2E(\xi\mu) - (E\mu)^2 = D\xi + D\mu$.

Определение. Квадратный корень из дисперсии называется средним квадратичным отклонением и используется для оценки масштаба возможного отклонения случайной величины.

6 Двоичный поиск и неравенство Крафта

Зададим используемые обозначения:

m — число элементов.

p_i — вероятность.

s_i — число битов и длина пути поиска.

Теорема. *Для того, чтобы набор целых чисел s_i, $i \in 1 : m$ мог быть набором длин путей поиска (префиксных двоичных кодов) в схеме с m исходами, необходимо и достаточно, чтобы*

$$\sum_{i \in 1:m} 2^{-s_i} \leq 1$$

Доказательство. **Необходимость**:

Рассмотрим поисковую схему, представляющую собой двоичное дерево T с m листьями. Сопоставим каждой вершине k этого дерева, находящейся на расстоянии t от корня число $a_k = 2^{-t}$. Для корня r_0 имеем $a_{r_0} = 1$, поэтому достаточно доказать, что

$$a_{r_0} \geq \sum_{k \in F} a_k$$

где F — множество листьев. Для каждого не-листа (то есть вершины, из которой хоть что-то выпадает) $k \in M \backslash F$ верно, что $a_k \geq \sum_{r \in next(k)} a_r$, где $next(k)$ — множество вершин, непосредственно выходящих из k. Это множество состоит из одной или двух вершин; если вершин 2 выполняется равенство, иначе — неравенство. Суммируя неравенства по $M \backslash F$, получаем

$$\sum_{M \backslash F} a_k \geq \sum_{M \backslash \{r_0\}} a_r$$

Вычтем из обоих сумм промежуточные вершины, получим

$$a_{r_0} \geq \sum_{F} a_k$$

Достаточность:

Возьмем m чисел, удовлетворяющих условию теоремы, расположим их в порядке возрастания. Определим $a_k = 2^{-s_k}$ и положим $b_1 = 0,\ b_{k+1} = b_k + a_k$. Рассмотрим последовательности из нулей и единиц t_k, являющиеся двоичными представлениями дробей b_k, в каждой дроби b_k возьмем первые s_k знаков. Покажем, что никакая из последовательностей t_k не является началом другой такой последовательности.

Так как длины последовательностей t_k не убывают с ростом k, достаточно доказать, что никакая t_k не является началом последовательности с большим номером. Дроби b_i возрастают, то есть $b_k < b_{k+1} < ...b_m \leq 1$.

	s_k	a_k	b_k	t_k
1	2	010000	000000	00
2	2	010000	010000	01
3	3	001000	100000	100
4	3	001000	101000	101
5	3	001000	110000	110
6	5	000010	111000	11100
7	5	000010	111010	11101
8	5	000010	111100	11110
9	6	000001	111110	111110
10	6	000001	111111	111111

Обозначим через $b_r^{(k)}$ число, получающееся из b_r отсечением первых s_k цифр. Для таких «урезаний» остается неравенство, теперь уже нестрогое $b_k < b_{k+1}^{(k)} < ... < b_m^{(k)}$. Следовательно, начало $t_r^{(k)}$ никакой дроби b_r не совпадает с t_k при $r > k$. □

Неравенство Крафта используется при алфавитном кодировании методом Фанно. Также имеются некие связи с алгоритмом Хаффмена.

7 Энтропия. Леммы. Теорема об энтропии

Определение. Энтропией случайной схемы называется мера содержащейся в этой схеме неопределенности. Она задается как вполне определенная функция от вероятностей исходов. Если мы имеем вероятностную схему $\mathscr{P}$ с m исходами и вероятности этих исходов равны $p_1, ..., p_m$, то энтропия этой схемы определяется как

$$H(\{p_1, ..., p_m\}) = \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$$

(что, конечно же, математически неверно, поскольку мы еще не умеем определять функции от переменного числа аргументов, однако следует её рассматривать как функцию $H(\mathscr{P})$).

Свойства энтропии:

1) Непрерывна на p_i;

2) При перестановке вероятностей $\{p_1, ..., p_m\}$ энтропия не меняется;

3) $H(\{\frac{1}{2}; \frac{1}{2}\}) = 1$ (энтропия от двух равновероятных событий равна 1).

4) При фиксированном m наибольшей неопределенностью обладает равновероятная схема (ну, это очевидно же :-)). Обозначим $H(\{\frac{1}{m}, ..., \frac{1}{m}\}) = h(m)$, где $h(m)$ — функция.

5) $h(m)$ возрастает с ростом m. Логично, что с ростом числа исходов неопределенность растет.

6) Рассмотрим схему $\mathscr{P}_m$ с m исходами и вероятностями $\{p_1, ..., p_m\}$ и схему $\mathscr{R}_k$ с k исходами и вероятностями $\{q_1, ..., q_k\}$. Образуем вероятностную схему с $m + k - 1$ исходами следующим образом: сначала выбирается исход из $\mathscr{P}_m$ и если это m-ый исход, то выбирается еще один исход из $\{q_1, ..., q_k\}$, остальные $m - 1$ исходов считаются окончательными. Тогда $H(\mathscr{P}\mathscr{R}) = H(\mathscr{P}_m) + p_m H(\mathscr{R}_k)$. ($p_m$ — это вероятность, если чо).

Теорема. *Единственная функция на множестве всех вероятностных схем, удовлетворяющая условиям 1-6 — функция H.*

Доказательство. Пусть функция G определена для всех вероятностных схем и удовлетворяет 1-6. Положим

$$G\left(\left\{\frac{1}{m}, .., \frac{1}{m}\right\}\right) = g(m)$$

Прервем доказательство и докажем две леммы, которые нам потребуются в дальнейшем:

Лемма. *(1)* $g(m) = \log_2 m$

Доказательство. Пусть имеются 2 схемы с равновероятностными исходами, $\mathscr{Q}_k$ и $\mathscr{Q}_l$. Составим из нее комбинированную схему $\mathscr{Q}_{kl}$ с kl равновероятными исходами: $(1,1), ..., (1,l), (2,1), ..., (2,l), ..., (k,1), ..., (k,l)$. Эта схема получается последовательным присоединением к схеме $\mathscr{Q}_k$ схемы $\mathscr{Q}_l$ k раз в продолжение схемы $\mathscr{Q}_k$. Из свойства 6 получаем $g(kl) = g(k) + g(l)$. Отсюда $g(m^k) = k \cdot g(m)$ и $g(2^k) = k$ в силу свойства 3. Теперь обозначим через s целую часть $\log_2 m^k$, ввиду монотонности g имеем $g(2^s) \le g(m^k) \le g(2^{s+1}) \Leftrightarrow s \le kg(m) \le s + 1$. Отсюда:

$$\frac{s}{k} \le g(m) \le \frac{s+1}{k} \Leftrightarrow 0 \le g(m) - \frac{[k \log_2(m)]}{k} = g(m) - \log_2 m + \frac{\{k \log_2 m\}}{k} \le \frac{1}{k}$$

Устремив $k \to \infty$ получим утверждение леммы. Используем участок

$$0 \le g(m) - \log_2 m + \underbrace{\frac{\{k \log_2 m\}}{k}}_{\to 0} \le \frac{1}{k}$$

□

Лемма. *(2) Если набор вероятностей $\{p_1, ..., p_m\}$ состоит из рациональных чисел, то $G(\{p_1, ..., p_m\}) = H(\{p_1, ..., p_m\})$.*

Доказательство. Пусть все $p_i = {}^{r_i}/_n$. Комбинируя схему $\mathscr{P}_m$ со схемами $\mathscr{Q}_{r_i}$, получим схему $\mathscr{Q}_n$ с n равновероятными исходами. Из свойства 6 $G(\mathscr{Q}_n) = G(\mathscr{P}_m) + \sum_{i \in 1:m} p_i G(\mathscr{Q}_{r_i})$, по лемме 1:

$$G(\mathscr{P}_m) = \log_2 n - \sum_{i \in 1:m} p_i \log_2 r_i = \sum_{i \in 1:m} p_i \log_2 \frac{1}{p_i}$$

□

Продолжим доказательство теоремы. Воспользуемся свойством непрерывности функции G. Для любого набора вероятностей $\{p_1, ..., p_m\}$ рассмотрим сходящуюся к нему подпоследовательность рациональных наборов $\{p_1^{(k)}, ..., p_m^{(k)}\}$. По лемме 2 для каждого из этих наборов $G = H$, так как функции непрерывны, то это равенство выполняется и в предельной точке. □

8 Код Шеннона-Фанно. Алгоритм Хаффмена

Определение. Префиксный код — код, любая последовательность которого не может являться началом другой последовательности. Это требование вводится для того, чтобы однозначно декодировать любую последовательность символов.

Если предположить, что кодируемые символы появляются в тексте независимо, то нужно стремиться уменьшать число битов на один символ, то есть математическое ожидание длины кодовой комбинации случайно выбранного символа, которое равно

$$\sigma = \sum_i p_i s_i$$

где p_i — вероятность, а s_i — длина кодовой последовательности i-того символа.
Таким образом, возникает

Проблема. о префиксном коде. Минимизировать математическое ожидание σ по всем наборам длин $\{s_i\}$, удовлетворяющим неравенству Крафта.

Шеннон и Фанно предложили строить код, близкий к оптимальному (т.е. с минимальным σ), следующим способом: разбить все символы на 2 группы с приблизительно равными суммарными вероятностями, коды первой группы начинать с нуля, а коды второй — с 1. Внутри каждой группы делать то же самое, пока в каждой группе не станет по 1 символу.

Элегантный алгоритм был предложен Хаффменом. Он основывается на следующих свойствах оптимального набора:

Лемма. *(1) Пусть $\{p_i\}$ — набор вероятностей символов и $\{s_i\}$ — длины оптимальных кодовых комбинаций. Если $p_1 \ge ... \ge p_n$, то $s_1 \le ... \le s_n$.*

Доказательство. Вытекает из решения задачи о перестановки, минимизирующей скалярное произведение двух векторов. (Теорема: наименьшее значение суммы попарных произведений достигается при сопоставлении возрастающей последовательности x_i и убывающей последовательности y_i). □

Лемма. *(2) В обозначениях леммы 1 две самые длинные кодовые комбинации имеют одинаковую длину, то есть $s_{n-1} = s_n$.*

Доказательство. Пусть $s_n > s_{n-1}$, следовательно, n-я кодовая комбинация — самая длинная. Так как никакая кодовая комбинация не является началом другой, то, сократив n-ю комбинацию до длины $n-1$-ой, мы получим вновь уникальную комбинацию, более короткую, чем раньше, что невозможно для оптимальной перестановки. □

Лемма. *(3) Рассмотрим наравне с нашей задачей P сокращенную задачу P', которая получается объединением двух самых редких символов в один. Это два последних символа с суммарной вероятностью $p'_{n-1} = p_{n-1} + p_n$. Минимальное значение целевой функции в задаче P' отличается от значения той же функции в задаче P на p'_{n-1}, а оптимальный кодовый набор задачи P получается из решения для задачи P' удлинением на один бит кодовой последовательности для объединенного символа.*

(то есть оптимальное решение первой задачи получается разделением последнего элемента второй задачи)

Доказательство. Действительно, каждому кодовому набору для задачи P' можно сопоставить кодовый набор для задачи P так, как указано в лемме. Теперь в задаче P найдем минимум на множестве кодов, который обозначим за C_n, а в задаче P' аналогичным образом найдем C_{n-1}. В соответствии с леммой 2 в задаче P можно искать минимум на множестве C'_n, в собственном подмножестве C_n, состоящем из наборов, в котором самые длинные кодовые последовательности имеют одинаковую длину. Имеется взаимно однозначное соответствие между C_{n-1} и C'_n, и значения целевых функций на соответствующих элементах двух множеств отличаются на постоянное слагаемое. Следовательно, минимумы отличаются на постоянное слагаемое, так что минимуму для задачи P' соответствует минимум на C'_n, а он будет минимумом для P. □

Алгоритм Хаффмена прост. Объединяем символы с наименьшими вероятностями в один со сложением вероятностей для получившегося символа. Произведем эти действия до получения 2-х «склеенных» символов. Затем обозначим один из них за 0, а другой за 1 и выполним обратный ход. Расщепим каждый символ на 2, одному присвоим значение 00, другому 01 (10 и 11 для второго символа). Выполнять рекуррентно до тех пор, пока каждый символ не обзаведется собственным кодом.

9 Метод Барроуза-Уилера

Будем объяснять этот алгоритм на большом примере. Возьмем для примера фразу

ГУСИ-ГУСИ-ГА-ГА-ГА$

Построим для нее суффиксное дерево по следующему алгоритму:

1) Выпишем в алфавитном порядке символы, составляющие фразу (в нашем случае $,-,А,Г,И,С,У).

2) Будем составлять дерево путем перечисления остатков всех букв по порядку: (открыть файл с хренью)

Теперь составим последовательность символов. Берем символ, предшествующий каждому суффиксу (т.е. берем символ, которым начинается ветвь суффикса (доводим до корешка, то есть до первого символа в дереве суффиксовСС) и смотрим ему предшествующий в фразе. Для суффикса -ГА$ таким символом будет А).

АААИИГГГ—-$ССУУГГ.

Теперь составляем табличку: каждый новый символ идет вперед; в строку «номер» записывается порядковый номер текущего символа в текущей строке (нумерация с единицы); добавление символа обнуляет счетчик.

Символ	Номер	Текущая строка
А	0	А
А	1	А
А	1	А
И	0	ИА
И	1	ИА
Г	0	ГИА
Г	1	ГИА
Г	1	ГИА
-	0	-ГИА
-	1	-ГИА
-	1	-ГИА
-	1	-ГИА
$	0	$-ГИА
С	0	С$-ГИА
С	1	С$-ГИА
У	0	УС$-ГИА
У	1	УС$-ГИА
Г	5	ГУС$-ИА
Г	1	ГУС$-ИА

В качестве шифра выступает последовательность номеров, а в качестве ключа — порядок появления символов в строке. В нашем случае:

0110101101110010151 — код, а порядок появления символов считывается по первому столбцу — АИГ-$СУ.

Теперь расшифруем нашу последовательность. Зная код, добавляем в начало последовательности букву, когда встречаем в коде очередной ноль; буква, на чей номер указывает цифра кода, переставляется в начало строки.

Номер	Текущая строка
0	А
1	А
1	А
0	ИА
1	ИА
0	ГИА
1	ГИА
1	ГИА
0	-ГИА
1	-ГИА
1	-ГИА
1	-ГИА
0	$-ГИА
0	С$-ГИА
1	С$-ГИА
0	УС$-ГИА
1	УС$-ГИА
5	ГУС$-ИА
1	ГУС$-ИА

(Таблица 1)

Теперь возьмем символы в алфавитном порядке и выпишем число символов в расшифровываемой

строке. Они легко получаются подсчетом количества того или иного символа на первом месте «Текущей строки».

$	-	А	Г	И	С	У
1	4	3	5	2	2	2

(Таблица 2)

Теперь составим еще одну таблицу, которая содержит в себе нашу полученную числовую последовательность и суммы соответствующих элементов этой и составляемой строки:

Буквы	$	-	А	Г	И	С	У
Готовая последовательность	1	4	3	5	2	2	2
Составляемая последовательность	1	2	6	9	14	16	18

(Таблица 3)

(складываются значения в столбиках и сумма записывается в нижнюю ячейку следующего столбца).

Теперь — еще одна таблица: в верхней строке просто нумерация; следующая строка — последовательность букв (просто считываем первые буквы Таблицы 1); третья строка — сопоставление «составляемой последовательности» и «букв» Таблицы (3), при использовании число инкрементируется.

Нумерация	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Буквенная посл.	А	А	А	И	И	Г	Г	Г	-	-	-	-	$	С	С	У	У	Г	Г
Числовая посл.	6	7	8	14	15	9	10	11	2	3	4	5	1	16	17	18	19	12	13

(Таблица 4)

И наконец, составим пятую таблицу по немного странному правилу:

Пишем в конец строки символ $. Затем смотрим сопоставляемую ему цифру из числовой последовательности, выбираем ту же цифру из нумерации и пишем соответствующую букву СЛЕВА от знака $: А$. И так далее.

10 Шифрование с открытым ключом

Определение. Алгоритмы шифрования с открытым ключом или антисимметричное шифрование строится на существовании пары ключей: публичного, для шифрования сообщения и приватного для его дешифровки.

Рассмотрим алгоритм шифрования RSA, который является одним из самых популярных алгоритмов шифрования. Для создания шифра выбирается большое число n — произведение двух простых множителей, p, q. При кодировании используется некое целое число e, а при декодировании нужно иметь другое число, d. Перейдем к деталям:

Пусть $n = p \cdot q$, где p, q — два простых числа. Положим $\varphi = (p-1)(q-1) = p \cdot q - p - q + 1$. Докажем две леммы:

Лемма. *Пусть a — число, взаимно простое с n. Тогда $a^{\varphi} = 1 \mod n$.*

Доказательство. (необязательное) Пусть $\Phi(n)$ — множество всех чисел, не превосходящих n и взаимно простых с ним. Заметим, что $|\Phi(n)| = \varphi$ (очевидно, так как не взаимно просты с n числа вида $p \cdot 1, p \cdot 2, ..., p \cdot q$ и аналогично с q. Так как само произведение входит в такое множество дважды, прибавим единицу). Умножение числа из этого множества на a по модулю n — биекция: если $b \cdot a = c \cdot a$, то, поскольку a взаимно просто с n, разность $c - b$ должна делиться на n. Таким образом,

$$\prod_{b \in \Phi(n)} b = \prod_{b \in \Phi(n)} (b \cdot a) = a^{\varphi} \prod_{b \in \Phi(n)} b \mod n$$

откуда верно утверждение леммы. □

Рассмотрим алгоритм шифрования:

Получатель:

1) Выбирает 2 больших простых числа p и q.

2) Перемножает $p \cdot q = n$, находит $\varphi(n) = (p-1)(q-1)$.
3) Выбирает число e, взаимно простое с $\varphi(n)$.
4) Определяет число d, такое, что $de = 1 \mod \varphi(n)$.
5) Отправляет отправителю открытые ключи n и e.

Отправитель:

1) Получает открытые ключи n и e.
2) Делит сообщение на кусочки, меньшие n.
3) Каждый кусочек x возводит в степень e: $x^e \mod n = c$.
4) Отправляет результат c получателю.

Дешифровка (получатель):

Возводит полученное число в степень d по модулю n: $c^d = x^{e^d} = x^{k\varphi(n)+1} \mod n \Leftrightarrow x^{k\varphi(n)} \cdot x \mod n = x$, расифровка получена.

На примере:

Зашифруем и расшифруем сообщение «ХУЙ» по алгоритму RSA. Для простоты возьмем небольшие числа - это сократит наши расчеты.

1) Выберем $p = 3,\ q = 11$.
2) $n = 33$, $\varphi(n) = 20$.
3) $e = 7$.
4) $de = 1 \mod 20 \Rightarrow d = 3$.
5) $n, e = \{33, 7\}$ — отправлены отправителю.

Шифрование сообщения:

1) Пусть Х=1, У=2, Й=3. Возводим в степень: $1^7 \mod 33 = 1,\ 2^7 \mod 33 = 29,\ 3^7 \mod 33 = 9$.
2) Пакет данных $\{1, 29, 9\}$ отправляется получателю.

Дешифровка:

1) Получатель возводит всё в степень d по модулю n: $1^3 \mod 33 = 1,\ 29^3 \mod 33 = 2,\ 9^3 \mod 33 = 3$.
2) Сообщение «ХУЙ» дешифровано.

11 Теорема о связном подграфе

Определение. Граф — тройка $\langle M, N, T \rangle$, где M, N — конечные подмножества, а T — отображение $T : N \to M \times M$ (ориентированный граф), где:

M — множество вершин, N — множество ребер.

$G = (M, N)$ — запись графа.

Определение. Путь длины k в ориентированном графе из вершины v в u — набор вершин $(v_0, ..., v_k),\ v_0 = v,\ v_k = u$ и $(v_{i-1}, v_i) \in N_{i \in [1,k]}$.

Определение. Простой путь — путь, в котором все вершины различны.

Определение. Контур в ориентированном графе — простой путь, у которого совпадают начальные и конечные вершины и который имеет хотя бы 1 ребро.

Определение. Цепь — аналог пути для неориентированного графа.

Определение. Цикл — простая замкнутая цепь.

Определение. Полный граф — граф, где все вершины соединены со всеми.

Определение. Граф связный, если любые две вершины соединены цепью.

Теорема. *$\langle M, N, T\rangle$ — связный. $\exists \langle M, N', T\rangle$, такой, что $|M| - 1 = |N'| = k$ (связный). При этом вершины можно пронумеровать от 0 до k, а дуги из N' от 1 до k, так, что $\forall u \in N'$ верно $num(u) = max\{num(beg(u)), num(end(u))\}$.*

Доказательство. По индукции. Будем строить подграф $\langle M', N', T\rangle$ в графе $\langle M, N, T\rangle$.

База: i_0 (только вершина). $k = 0$.

ИП: Пусть верно для $\langle M', N', T\rangle$, причем $M \backslash M' \neq \varnothing$. Выберем какую-либо вершину $\bar{i} \notin M'$. Так как граф $\langle M, N, T\rangle$ связен, то существует цепь из $i_0 \to \bar{i}$. $i_0 \in M', \bar{i} \notin M' \Rightarrow \exists i', i''$, соединенные ребром, таким что $i' \in M'$, $i'' \notin M'$. Добавим i'' в M', а дугу, их соединившую, в N'.

Количество дуг в N $k := k + 1$. Количество вершин в первоначальном графе уменьшается с каждой итерацией, следовательно, процесс конечен. □

Следствие 1: Если $|N| < |M| - 1$, то граф несвязен.

Следствие 2: Если $|N| > |M| - 1$, то граф содержит циклы.

Определение. Граф, удовлетворяющий теореме, называется деревом.

12 Деревья. Теорема о 6 эквивалентных определениях дерева

Определение. Максимальный граф — граф, в котором добавление дуги между любыми вершинами спровоцирует появление цикла.

Теорема. *Следующие определения дерева эквивалентны:*

1) Связный граф без цикла;

2) Связный граф, в котором число_дуг=число_вершин-1;

3) Граф без цикла, в котором число_дуг=число_вершин-1

4) Минимальный связный граф;

5) Максимальный граф без цикла;

6) Граф, в котором любые 2 вершины соединены единственной цепью;

Доказательство.

$1 \Rightarrow 5$) Докажем, что если граф без циклов связен, то это максимальный граф без циклов. Действительно, при добавлении дуги к графу замыкается цепь

$5 \Rightarrow 6$) Если при добавлении дуги появляется цикл, значит, эта новая дуга замкнула цепь, соединяющую её начало и конец. Значит, любые 2 вершины соединены цепью.

$6 \Rightarrow 4$) Связность графа очевидна. Так как для любых двух вершин соединяющая их цепь единственна, для начала и конца любой дуги этот путь — именно сама эта дуга. Её удаление разрывает цепь между ними.

$4 \Rightarrow 2$) Нужно доказать, что если граф минимально связный, то в нем число дуг n на 1 меньше, чем число вершин m. Воспользуемся следствием теоремы об остовном дереве. Для связности требуется $n \geq m - 1$, но если $n > m - 1$, то в графе найдется дуга, удаление которой не скажется на остовном дереве, то есть её удаление не нарушит связности, следовательно, эта дуга замыкает цикл.

$2 \Rightarrow 3$) Докажем, что при $n = m - 1$ связный граф не имеет циклов. Пронумеруем все вершины и дуги. Вершина с наибольшим номером инцидентна двум дугам, для каждой из которых этот номер — максимум из номера начала и номера конца, следовательно, равен номеру дуги. Но две дуги не могут иметь один и тот же номер.

$3 \Rightarrow 1$) Если в графе без циклов $n = m - 1$, то он связен. Действительно, если он не связен, то в нем есть компонента связности, в которой найдется цикл. □

13 Задача о кратчайшем остовном дереве. Алгоритм Прима

Определение. Дерево, являющееся частичным графом связного графа — остовное.

Проблема. (о кратчайшем дереве в графе):

Пусть каждой дуге j графа $\langle M, N, T\rangle$ сопоставляется неотрицательное число $l[j]$, которое называется длиной этой дуги. Требуется построить такое остовное дерево, у которого сумма длин дуг минимальна.

Алгоритм Прима.

Начальная точка алгоритма — любая вершина графа. Алгоритм «помечает» вершины графа, которые добавлены к остовному дереву. Назовем каймой набор дуг, у которых одна вершина помечена, а другая нет. Алгоритм выбирает из каймы дугу с наименьшей длиной и помечает её второй конец. Кайма меняется, алгоритм повторяет действия до тех пор, пока есть непомеченные вершины.

Теорема. *Алгоритм Прима строит наименьшее остовное дерево.*

Доказательство. Пусть граф G был связным, т.е. ответ существует. Обозначим через T остов, найденный алгоритмом Прима, а через S — минимальный остов. Очевидно, что T действительно является остовом (т.е. поддеревом графа G). Покажем, что веса S и T совпадают.

Рассмотрим первый момент времени, когда в T происходило добавление ребра, не входящего в оптимальный остов S. Обозначим это ребро через e, концы его — через a и b, а множество входящих на тот момент в остов вершин — через V (согласно алгоритму, $a \in V$, $b \notin V$, либо наоборот). В оптимальном остове S вершины a и b соединяются каким-то путём p; найдём в этом пути любое ребро g, один конец которого лежит в V, а другой — нет. Поскольку алгоритм Прима выбрал ребро e вместо ребра g, то это значит, что вес ребра g больше либо равен весу ребра e.

Удалим теперь из S ребро g, и добавим ребро e. По только что сказанному, вес остова в результате не мог увеличиться (уменьшиться он тоже не мог, поскольку S было оптимальным). Кроме того, S не перестало быть остовом (в том, что связность не нарушилась, нетрудно убедиться: мы замкнули путь p в цикл, и потом удалили из этого цикла одно ребро).

Итак, мы показали, что можно выбрать оптимальный остов S таким образом, что он будет включать ребро e. Повторяя эту процедуру необходимое число раз, мы получаем, что можно выбрать оптимальный остов S так, чтобы он совпадал с T. Следовательно, вес построенного алгоритмом Прима T минимален, что и требовалось доказать. □

14 Алгоритм Краскала

Алгоритм Краскала:

1) Сортируем дуги графа по длине;

2) Строим дерево — множество всех дуг. На начальном шаге оно пусто.

3) Рассматриваем наименьшую дугу. Проверяем, принадлежат ли её концы одному и тому же множеству вершин. Если это так, пропускаем дугу. Иначе добавляем её и её концы в дерево (т.е. концы идут в множество вершин, либо новое, если оба конца не принадлежат никакому множеству, либо в одно из существующих, если один из концов принадлежит какому-либо множеству вершин. Если концы дуги принадлежат разным множествам, то необходимо объединить их в одно). Алгоритм завершает работу, если просмотрены все дуги.

Теорема. *Алгоритм Краскала строит наименьшее остовное дерево.*

Доказательство. Пусть $\langle M, N^*\rangle$ — дерево, построенное алгоритмом Краскала, $\langle M, N_{opt}\rangle$ — оптимальное дерево. Первые k длин ребер совпадают у обоих деревьев: $q_1, ..., q_k$ (если мы их упорядочим по возрастанию, конечно же). Вставим элемент $*$ из N^* в N_{opt} на то же место. Появляется цикл. Заметим, что

с $q_1, ..., q_k$ $*$ циклов не образовывал, иначе в N^* были бы циклы, следовательно, $*$ образовывает цикл с чем-то из правой части N_{opt} : пусть с q_i, где $i > k+1$. Удалим q_i, при этом $l[q_i] \geq l[q_j]$, где $i \in 1:k$. Следовательно, длина N^* не изменилась, длина N_{opt} либо не изменилась, либо уменьшилась; в первом случае сведем N_{opt} к N^*, во втором N_{opt} не является максимальным. □

15 Задача о кратчайшем пути. Алгоритм Дейкстры

Проблема. Задан граф $\langle M, N\rangle$ и каждой его дуге сопоставлено положительное число $l[u]$. Для двух заданных вершин i^- и i^+ требуется найти путь минимальной длины с началом i^- и концом i^+.

Алгоритм Дейкстры:
Зададим:
M_0 — вершины, расстояние до которых уже вычислено.
M_1 — вершины, расстояние до которых вычисляется.
M_2 — вершины, расстояние до которых еще не вычислено.
Начальное состояние:
$M_0 = \varnothing,\ M_1 = \{i^-\},\ M_2 = \{M \backslash i^-\}$.
Стандартный шаг алгоритма:
Берем вершину из множества M_1 (верхний элемент из очереди). Пусть v — выбранная вершина. Переводим эту вершину во множество M_0. Затем просматриваем все ребра, выходящие из этой вершины. Пусть t — второй конец этого ребра, а l — длина ребра. D_r — длина пути до вершины r
$\star$) Если $t \in M_2$, то переносим t в M_1, в конец очереди, а $D_t = D_v + l$.
$\star$) Если $t \in M_1$, то $D_t = \min(D_t, D_v + l)$. Сама t никак не передвигается в очереди.
$\star$) Если $t \in M_0$ и если D_t можно улучшить ($D_t > D_v + l$), то улучшаем D_t, а вершину t возвращаем в M_1 в начало очереди.

Теорема. *Алгоритм Дейкстры находит путь минимальной длины.*

Доказательство. 1) $\forall v \in M_0,\ f(v) = \delta(x_0, v)$ — длина кратчайшего пути от x_0 до v, и существует путь из x_0 в v целиком лежащий в M_0.
2) $v \in M \backslash M_0,\ f(v)$ — длина наименьшего пути от x_0 до v, такого, что все вершины пути принадлежат M_0 кроме v.
Обоснуем выполнение утверждения 1:
$\exists y : f[y] = \min_{i \in M_1} f[i]$. $f[y]$ — конечное значение. От противного: $\exists$ путь из x_0 в y, с длиной $l(p)$. Тогда $l(x_0, y) > l(x_0, z)$, где $z \in p$ и $l(x_0, y) > l(x_0, z) \geq f(z) \geq f(y)$. Противоречие.
Обоснуем выполнение утверждения 2:
$M_0' = M_0 \cup \{y\}$. $f'(M) = \min\{f(V), f(y) + l(y, v)\}$. □

16 Задача о максимальном парасочетании в графе. Алгоритм построения

Определение. Ориентированный граф $\langle M, N\rangle$ — двудольный, если множество его вершин разбито на два множества $M_b,\ M_e$ и все начала дуг принадлежат M_b, а концы — M_e.

Определение. Набор дуг $J \subset N$ называется парасочетанием, если $\forall j_1, j_2 \in J,\ j_1 \neq j_2$, начала и концы этих дуг различны $beg(j_1) \neq beg(j_2)$ и $end(j_1) = end(j_2)$.

Посмотрим, каким образом можно построить парасочетание, максимальное по числу входящих в него дуг. Вместе с парасочетанием будем строить некое специальное множество вершин.

Определение. Множество вершин M' контролирует дугу j, если если начало или конец этой дуги принадлежит M'. Нас интересует контрольное множество, в котором для любой дуги j найдется контролирующая её вершина. Если размер парасочетания мы хотим максимизировать, то контрольное множество, очевидно, должно быть минимальным. Недостроенное контрольное множество будем называть тестовым.

Алгоритм построения максимального парасочетания.

Обозначим через $beg(K) = \{beg(j)|j \in K\},\ end(K) = \{end(j)|j \in K\}$, где K — любое множество дуг.

Состояние вычислительного процесса. На каждом шаге алгоритма имеется текущее парасочетание N', разбитое на два подмножества $N' = N'_b \cup N'_e$ (по положению контролирующей точки — в начале или в конце дуги). Объединение M' множеств $M'_b = beg(N'_b)$ и $M'_e = end(N'_e)$ составляет текущее тестовое множество. Каждой дуге $j \in N'$ сопоставим дугу $\delta(j)$ — её дублер, который не входит в парасочетание и имеет тот же конец, что и j. Припишем каждой дуге из N'_e положительный ранг $\rho(j)$.

Начальное состояние: Первоначально множество N' пусто, соответственно, пусты и N'_b, N'_e, M'.

Стандартный шаг: Если все дуги из N контролируются множеством M', работа алгоритма завершается.

Пусть существует дуга j_0, не контролируемая множеством M', так что $beg(j_0) \notin beg(N'_b),\ end(j_0) \notin end(N'_e)$ (то есть контролирующей точки для этой дуги нет). Возможны четыре случая:

1) $beg(j_0) \notin beg(N'),\ end(j_0) \notin end(N')$. Мы нашли дугу с началом и концом, не встречающимися в текущем парасочетании. Эту дугу можно добавить к парасочетанию: $N' := N' \cup \{j_0\},\ N'_b := N',\ N'_e := \varnothing,\ M' := beg(N')$.

2) $beg(j_0) \notin beg(N'),\ end(j_0) \in end(N')$. Так как $end(j_0) \notin end(N'_e)$, но $end(j_0) \in end(N')$, то $\exists j_1 \in N'_b$, такая, что $end(j_0) = end(j_1)$. Переведем дугу j_1 из N'_b в N'_e, назначив ей дублером j_0 и приписав им ранг 1.

3) $beg(j_0) \in beg(N'),\ end(j_0) \in end(N')$. Существует такая дуга $j_1 \in N'_b$, что $end(j_0) = end(j_1)$ и существует дуга $j_2 \in N'_e$, такая, что $beg(j_0) = beg(j_2)$. Переведем дугу j_1 из N'_b в N'_e, назначив ей дублером j_0 и приписав им ранг $\rho(j_2) + 1$. Отметим, что так наращиваются постепенно цепочки дуг и дублеров и каждая кончается парой дуг первого ранга.

4) $beg(j_0) \in beg(N'),\ end(j_0) \notin end(N')$. Существует дуга $j_2 \in N'_e$, такая, что $beg(j_0) = beg(j_2)$. Присоединение j_0 к цепочке, начинавшейся с j_2 дает нам цепочку с нечетным числом дуг. В нем дуг из парасочетания на одну меньше, чем дублеров, при этом начала и концы дублеров — те же вершины, что и у дуг парасочетания, а начало дублера первого ранга и конец дуги j_0 в парасочетании не встречались. Заменим дуги парасочетания их дублерами. Размер множества N' вырастет на 1 и получится новое парасочетание. Переведем все дуги парасочетания из N'_e в N'_b.

17 Теорема Кенига

Теорема. *Размерность максимального парасочетания в двудольном графе равна минимальному размеру контрольного множества.*

Доказательство. Размер любого парасочетания не превосходит размера любого контрольного множества, так как каждая дуга парасочетания контролируется отдельной вершиной.

Алгоритм нахождения максимального парасочетания заканчивается, когда все дуги контролируются. При этом количество контролирующих вершин равно количеству дуг, входящих в максимальное парасочетание N. Следовательно, размер контрольного множества должен быть не меньше N. □

18 Алгоритм построения контролирующего множества

А здесь мы смотрим вот этот алгоритм 16.

19 хуй

20 хуй

21 хуй

22 хуй

23 Алгоритмы с гарантированной оценкой точности. Алгоритм Эйлера

Определение. Алгоритмы с гарантированной оценкой точности — алгоритм, для которого $\exists k : \frac{fA}{fopt} < k$, где fA — результат алгоритма, $fopt$ — оптимальный результат.

Определение. Эйлеров путь — путь, который проходит по каждому ребру, причем только один раз.

Определение. Эйлеров цикл — замкнутый эйлеров путь.

Определение. Степень вершины — число ребер, которые исходят из данной вершины.

Алгоритм Эйлера:
G — граф.
1) Строим минимальное остовное дерево T в G.
2) Удваиваем ребра в T.
3) С помощью поиска в глубину ищем эйлеров цикл: $1 \to 5 \to 2 \to 5 \to 3 \to 4 \to 3 \to 6 \to 3 \to 2 \to 1$.
4) Убираем из цикла повторяющиеся вершины: $1 \to 2 \to 5 \to 3 \to 4 \to 6 \to 1$.

Теорема. *Для алгоритма Эйлера* $k = 2$.

Доказательство. Пусть T — суммарный вес ребер в дереве T. $fopt \geq T$. $fA \leq 2 \cdot T$ (т.к. путь будет меньше, чем обходить все дерево T). Отсюда $\frac{f_A}{f_{opt}} \leq 2$. □

24 Алгоритм Кристофидеса

1) Строим минимальное дерево.

2) У вершин с нечетной степенью находим минимальное взвешенное парасочетание (среди ребер, соединяющих только эти вершины, находим такие, которые были бы паросочетанием и имели наименьшую длину). Добавляем дуги из него в дерево.

3) Дальше по алгоритму Эйлера.

25 хуй

26 хуй

Рис. 1: