

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

Опр

Пустое множество (\emptyset) - мн-во, которому \notin ни один элемент

Опр

Число элементов мн-ва A - мощность $|A|$

Опр

Множество чисел от k до l обозначается $k : l$

Опр

Мн-во A - подмн-во мн-ва B ($A \subset B$), если каждый элемент из A принадлежит B

Опр

С - объединение A и B ($A \cap B$), если оно состоит из всех элементов A и B ($C = \{x | x \in A \text{ и } x \in B\}$)

Опр

$\bigcap_{i=1}^n A_i$, $\bigcup_{i=1}^n A_i$ - объединение и пересечение конечного числа мн-в

$(\bigcap_{i \in I} A_i, \bigcup_{i \in I} A_i)$ - аналогично

Опр

Если пересечение мн-в пусто, то они называются дизъюнктивными

Опр

Мн-во C называется разностью мн-в A и B ($C = A \setminus B$), если оно состоит из всех эл-в, принадлежащих A и не принадлежащих B

Опр

$A \Delta B = A \setminus B \cap B \setminus A$ - симметрическая разность

Опр

Мн-во упорядоченных пар (i, j) , где $i \in A$, $j \in B$ называется прямым произведением мн-в A и B

$A \times B = \{(i, j) | i \in A, j \in B\}$

Замечание

Мощность прямого произведения $|A \times B| = |A| \cdot |B|$. Аналогично произведение \forall конечного числа множеств

Опр

Пусть A_1, \dots, A_k - ненулевые и попарно дизъюнктивные, $M = A_1 \cap \dots \cap A_k$ и мн-во $\{A_1, \dots, A_k\}$ называется разбиением М
(если они попарно не дизъюнктивные, то это покрытие)

Опр

Разбиение А мн-ва М называется измельчением В, если $\forall A_i \in A$ содержится в некотором $B_i \in B$

Опр

Пусть А, В - размельчения мн-ва М, разбиение С называется произведением А и В, если оно является из измельчением, причем самым крупным $C = A \cdot B$

Теорема

Произведение двух разбиений существует

Док-во

Предъявим разбиение, которое будет пересечением $A = \{A_1, \dots, A_k\}$ и $B = \{B_1, \dots, B_l\}$, точнее $D_{ij} = A_i \cup B_j$, $i \leq k$, $j \leq l$ и $\mathcal{P} = \cup D_{ij}$ (т.е. без пустых строк). Покажем, что тогда оно самое крупное.

Пусть $\exists F = \{F_1, \dots, F_t\}$ - измельчение А и В, тогда $\forall F_k \exists A_{i_k}, B_{i_k} : F_k A_{i_k}, B_{i_k} \Rightarrow F_k \subset (A_{i_k} \cup B_{i_k}) = D_{i_k j_k} \Rightarrow$ мельче F

2 Вектора из нулей и единиц

Пусть мн-во B состоит из двух элементов которые отождествляются с 0 и 1, т.е. $B = 0 : 1$

Произведение m экземпляров такого мн-ва обозначим за $B^m = (0 : 1)^m$, состоит из 2^m эл-ов

Опр

Вектор из нулей и единиц - упорядоченный набор из фиксированного числа нулей и единиц, т.е. эл-т мн-ва B^m

Упорядоченный набор из чисел обычно называется вектором, m - раз мерностью вектора, каждый отдельный элемент набора - компонента вектора

Замечание

Модели, в которых используются наборы из 0 и 1:

1. Геометрическая интерпретация

Точкой в m -мерном пространстве является m -мерный вектор, каждая его компонента - одна из декартовых координат точки. Набор из 0 и 1, рассматриваемый как точка в пространстве, определяет вершину куба, построенного на ортах (единичных отрезках) координатных вероятностей

2. Логическая интерпретация

Операции над векторами выполняются покомпонентно, т.е. независимо над соотв. компонентами векторов-операндов

Пример

x	0	0	0	1	1
y	1	1	1	0	1
$x \wedge y$	0	0	0	0	1
$x \vee y$	1	1	1	1	1
$x \equiv y$	0	0	0	0	1
$x \neq y$	1	1	1	1	0

3. Двоичное представление (натуральные числа)

Число представляется в виде суммы степеней 2

4. Состояние памяти компьютера

5. Сообщение, передаваемое по каналу связи

6. Можно задавать подмножества мн-ва $1 : n$

3 Алгоритм перебора 0-1 векторов. Коды Грэя

Опр

Код Грэя — такое упорядочение k -ичных (обычно двоичных) векторов, что соседние вектора отличаются только в одном разряде

Алгоритм

it - номер итерации, k_{it} - номер обновляемой компоненты

x_4	x_3	x_2	x_1	it	k_{it}
0	0	0	0	0	1
0	0	<u>0</u>	1	1	2
0	0	1	1	2	1
0	<u>0</u>	1	0	3	3
0	1	1	0	4	1
0	1	<u>1</u>	1	5	2
0	1	0	1	6	1
0	1	0	0	7	4
...		...			

Суть алгоритма: зафиксируем нулевое значение у m -й компоненты и переберем все наборы длины $m - 1$ для ост. компонент. Перебрав их меняем значение m -й компоненты на 1 и перебираем набор длины $m - 1$ в обратном порядке

Замечание

Явная формула для проверки $G_i = i \oplus (\lfloor i/2 \rfloor)$

4 Перебор элементов прямого произведения множеств

$$M(1:k) = M_1 \times M_2 \times \dots \times M_k$$

$$|M_1 \times M_2 \times \dots \times M_k| = \prod_{i \in 1:k} m_i, \text{ где } m_i = |M_i|$$

Пусть каждое M_i состоит из целых чисел от 0 до $m_i - 1$, тогда каждый элемент $M(1:k)$ - последовательность неотрицательных чисел r_1, \dots, r_k , причем $r_i < m_i$

$$\text{num}(r_1, \dots, r_k) = \sum_{i=0}^k r_i \cdot (\prod_{j=1}^{i-1} m_j) = r_1 + r_2 m_1 + \dots + r_k m_1 \cdot \dots \cdot m_{k-1}$$

5 Размещения, сочетания, перестановки без повторений

Опр

Перестановка из n без повторений - упорядоченный набор из n неповторяющихся элементов, каждый из которых берется из диапазона $1 : n$

$$|P_k| = n!$$

Опр

Размещение - упорядоченный набор из k неповторяющихся элементов из диапазона $1 : n$

$$A_n^k = \frac{n!}{(n-k)!} = n(n-1)(n-k+1)$$

Опр

Сочетание - набор из k неповторяющихся элементов из диапазона $1 : n$ (порядок не важен)

$$|C_n^k| = \frac{n!}{(n-k)!k!}$$

6 Размещения, сочетания, перестановки с повторениями

Перестановки с повторениями:

Последовательность длины n , составленных из k разных символов, i -ый из которых повторяется n_i раз ($n_1 + n_2 + \dots + n_k = n$)

Пример (aabc)

Перестановки: $abac, baac, aabc, aacb, abca, bac a, acab, bcaa, cbaa, cab a, caa b$

Число перестановок с повторениями длины из k разных элементов взятых соответственно по n_1, \dots, n_k раз каждый обозначается

$$P(n_1, n_2, \dots, n_k) = \frac{n!}{n_1!n_2!\dots n_k!}$$

7 Два алгоритма перебора перестановок. Нумерация перестановок

$$|P_k| = k! = |T_k|$$

P_k - мн-во всех перестановок

T_k - произведение к любых таких множеств M_i , каждый из которых представляет собой мн-во чисел от 0 до $i - 1$

$$T_k = \{0\} \times \{0, 1\} \times \dots \times \{0, 1, \dots, k - 1\}$$

Построим взаимно однозначное соответствие между P_k и T_k . Возьмем перестановку (t_1, \dots, t_k) следующим образом: для любого $i \in 1 : k$ найдем число значений, меньше r_i среди r_{i+1}, \dots, r_k - это число мы и примем в качестве t_i

В соответствии с таким определением чисел t_i в мн-ве T_k будет соответственно ??? значения m_i не возраст., а убывающая до единицы

Пример (4, 8, 1, 5, 7, 2, 3, 6)

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

По (t_1, \dots, t_k) легко восстановить исходную перестановку. Для этого меняя i от 1 до k нужно проверить мн-во значений S_i , которые могут быть в перестановке на i месте. Для $i = 1$ $S_1 = 1 : 8$, $t_1 = 3 \Rightarrow r_1 = 4$, далее $S_2 = 1 : 3 \cap 5 : 8$, $t_2 = 6 \Rightarrow r_2 = 8$. Если использовать это отображение при переборе, то перестановки будут перебираться в лексикографическом порядке

Опр

(r_1, \dots, r_k) предшествует (R_1, \dots, R_k) , если начала перестановок совпадают до индекса d , а дальше $r_d < R_d$

Утв

Из этого перестановки перебираются в лексикографическом порядке, можно вывести правило получения следующего:

1. В (r_1, \dots, r_k) найти наибольший суффикс (r_t, \dots, r_k) , в котором $r_t > \dots > r_k$ ($r_{i-1} < r_t$)

2. Выбрать (r_t, \dots, r_k) элемент следующий по величине после r_{t-1} , поставить после в возр. порядке

num	t _k				p _k			
0	0	0	0	0	1	2	3	4
1	0	0	1	0	1	2	4	3
2	0	1	0	0	1	3	2	4
3	0	1	1	0	1	3	4	2
4	0	2	0	0	1	4	2	3
5	0	2	1	0	1	4	3	2

Ещё один алгоритм

8 Задача о минимуме скалярного произведения

Пусть заданы числа x_1, \dots, x_m и y_1, \dots, y_m . Составим пары (x, y) , включив каждое x_i и y_i ровно в одну пару. Затем перемножим числа каждой пары и сложим полученное произведение. Требуется найти \min такое разбиение чисел на пары S

Теорема

$$\bar{x} = (x_1, \dots, x_n) \quad x_1 \geq x_2 \dots \geq x_n$$

$$\bar{y} = (y_1, \dots, y_n) \quad y_1 \leq y_2 \dots \leq y_n$$

$$S = \sum_{i=1}^n x_i y_i \rightarrow \min$$

Док-во

Покажем, что если найдутся пары чисел (x_i, y_i) и (x_j, y_j) : $x_i < x_j$, $y_i < y_j$, то S можно уменьшить, заменив парами (x_i, y_j) и (x_j, y_i)

Действительно,

9 Числа Фибоначчи. Теорема о представлении

Опр

Последовательность чисел Фибоначчи F:

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}, \quad n > 1$$

Утв

$$\varphi_n = \frac{F_{n+1}}{F_n} - \text{сходится}$$

Следствие

$$\varphi_n = \frac{F_{n+1}}{F_n} = \frac{F_{n-1} + F_n}{F_n} = 1 + \frac{1}{\varphi}$$

$$\Rightarrow \varphi = \frac{\sqrt{5} + 1}{2}$$

Лемма

При $n > 1$ выполнено $\varphi^{n+2} = \varphi^{n+1} + \varphi^n$

Док-во

Лемма

При $k > 2$ выполнено:

$$F_{2k} = F_{2k-1} + F_{2k-3} + \dots + F_1$$

$$F_{2k+1} = 1 + F_{2k} + F_{2k-2} + \dots + F_0$$

Док-во (по индукции)

$(k = 3)$:

$$F_6 = 8 = 5 + 2 + 1$$

$$F_7 = 13 = 1 + 8 + 3 + 1 + 0$$

$(k \rightarrow k + 1)$:

$$F_{2(k+1)} = F_{2k+2} = F_{2k+1} + F_{2k} = F_{2k+1} + F_{2k-1} + \dots + F_1 = ???$$

Теорема

Любое натуральное число можно однозначно представить в виде суммы чисел Фибоначчи

$$s = F_{i_0} + F_{i_1} + \dots + F_{i+r}, \text{ где } i_{k-1} + 1 < i_k, \quad k \in 1 : r \quad i_0 = 0$$

Док-во

Существование:

Пусть $j(s)$ - номер максимального числа Фибоначчи, не превосходящего s . Положим $s' = s - F_{j(s)}$. Из определения $j(s)$ следует, что $s' < F_{j(s)-1}$, иначе число Фибоначчи не было бы максимальным. Теперь мы получим искобое представление для s как представление s' , дополненное слагаемым $F_{j(s)}$

Единственность:

Пусть есть ёщё одно представление $s = F_{j_0} + \dots + F_{j_q}$. Н.У.О. считаем, что $j_q < j(s)$. Если мы заменим F_{j_q} на $F_{j(q)-1}$, то правая часть разве что лишь увеличится. Аналогично заменим с возможным увеличением предпоследнее слагаемое на $F_{j(s)-3}$. ???

10 Перебор сочетаний. Нумерация сочетаний

Состояние вычислительного процесса. Массив (x_1, \dots, x_m) номеров, включенных в сочетание. Начальное состояние: принять $x_i = i \quad \forall i \in 1 : m$. Стандартный шаг: просматривать компоненты вектора x , начиная с x_m и искать первую компоненту, которую можно увеличить (нельзя $x_m = n, x_{m-1} = n - 1$ и т.д.). Если такой нет, то закончить процесс. В противном случае пусть k - наибольшее число, для которого $x_k < n - m + k$, тогда увеличить x на единицу, а для всех следующих за k -ым продолжаем, но ряд от значения x_k , т.е. $x_i = x_k + (i - k)$

num	Сочетание					k
1	1	2	3	4	5	5
2	1	2	3	4	6	5
3	1	2	3	4	7	5
4	1	2	3	5	6	4
5	1	2	3	5	7	5

Удобно использовать вектора из 0 и 1, чтобы перенумеровать. С каждым сочетанием из n по m можно связать вектор из n нулей и единицЮ в крирпм единиц ровно m - числа, входящие в данное сочетание просто задают номера этих единиц

$$\text{num}(b[1 : n], m) = \begin{cases} \text{num}(b[1 : n], n) & b[n] = 0 \\ l_{n-1}^m + \text{num}(b[1 : n], m-1) & b[n] = 1 \end{cases}$$

Пример

$$\begin{aligned} \text{num}((0, 1, 0, 1, 0, 0, 1), 3) &= \\ &= C_6^3 + \text{num}((0, 1, 0, 1, 0, 0), 2) = C_6^3 + C_3^2 + \text{num}((0, 1, 0), 1) = \\ &= C_6^3 + C_3^2 + \text{num}((0, 1), 1) = C_6^3 + C_3^2 + C_1^1 + \text{num}((0), 0) = 24 \end{aligned}$$

11 Бином Ньютона и его комбинаторное использование

Треугольник Паскаля (в узлах C_n^k):

	1							
$n = 0$		1	1					
$n = 1$		1	2	1				
$n = 2$		1	3	3	1			
$n = 3$		1	4	6	4	1		
$n = 4$		1	5	10	10	5	1	
$n = 5$		1	6	15	20	15	6	1
$n = 6$		0	1	2	3	4	5	6

Опр

Бином Ньютона: $(a + b)^n = \sum_{k=0}^n C_n^k a^k b^{n-k}$

Лемма

$$C_{n-1}^{k-1} + C_{n-1}^k = C_n^k$$

Док-во (по индукции)

$$\begin{aligned} (a+b)^n &= a(a+b)^{n-1} + b(a+b)^{n-1} = \\ &= \sum_{k=0}^{n-1} C_{n-1}^k a^{k+1} b^{(n-1)-k} + \sum_{k=0}^{n-1} C_{n-1}^k a^k b^{1+(n-1)-k} = \\ &= \sum_{k=1}^n C_{n-1}^{k-1} a^k b^{n-k} + \sum_{k=0}^{n-1} C_{n-1}^k a^k b^{n-k} = \sum_{k=0}^n (C_{n-1}^{k-1} + C_{n-1}^k) a^k b^{n-k} \end{aligned}$$

Следствие

$a = 1, \quad b = 1:$

$$\sum_{k=0}^n C_n^k = 2^n$$

$a = 1, \quad b = -1:$

$$\sum_{k=0}^n C_n^k (-1)^k = 0$$

(благодаря $C_n^k = C_n^{n-k}$)

$a = 1, \quad b = i:$

$$\sum_{k=0}^n C_n^k i^k = (1+i)^n = (\sqrt{2} \cdot \cos \frac{\pi}{4} + i \sin \frac{\pi}{4})^n = 2^{\frac{n}{2}} \cdot e^{in\frac{\pi}{4}}$$

12 Свойства биномиальных коэффициентов

1. $C_n^k = C_n^{n-k}$
2. $C_{n-1}^{k-1} + C_{n-1}^k = C_n^k$
3. $C_n^m C_m^k = C_n^k C_{n-k}^{m-k}$

13 Основные определения теории вероятностей

14 Условные вероятности и формула Байеса

Опр (Условная вероятность)

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

Вероятность события B , если произошло A (все исходы, при которых произошли B и A на исходы с A)

Опр (Формула полной вероятности)

$$S_1, \dots, S_n \quad S_i \cap S_j = \emptyset$$

$$P(A) = \sum_{i=1}^n P(A \cap S_i) = \sum_{i=1}^n P(A | S_i) \cdot P(S_i)$$

Опр (Формула Байеса)

$$P(S_i | A) = \frac{P(A | S_i)P(S_i)}{\sum_{i=1}^n P(A | S_i) \cdot P(S_i)} \quad \text{по Григорьевой}$$

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

15 Математическое ожидание и дисперсия случайной величины

Опр

Случайная величина - числовая функция $\alpha : U \rightarrow \mathbb{R}$ на вероятностном пр-ве.

hint: U - мн-во событий

Опр

$$E(\alpha) = \sum_{u \in U} \alpha(u) \Pr(u) \text{ - мат. ожидание случайной величины } \alpha$$

$\Pr(u)$ - условная вероятность события u

Свойства

1. Если $\Pr(u) = 1 \Rightarrow E(\alpha) = \alpha(u)$
2. α, β - случ. вел. $E(\alpha + \beta) = E(\alpha) + E(\beta)$ (Линейность)
3. Если $\alpha = c\beta$, где $c = const$ $E(\alpha) = cE(\beta)$
4. $E(\alpha \cdot \beta) = E(\alpha) \cdot E(\beta)$ если α и β нез.

Док-во

Линейность

$$E(\alpha + \beta) = \sum_{u \in U} (\alpha(u) + \beta(u)) \Pr(u) = \sum_{u \in U} \alpha(u) \Pr(u) + \sum_{u \in U} \beta(u) \Pr(u) = E(\alpha) + E(\beta)$$

Опр

Мат. ожидание квадрата отклонения случайной величины от ее мат. ожидания называется дисперсией этой случайно величины

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

Дисперсия характеризует разброс случайной величины вокруг ее мат. ожидания

$$\begin{aligned} D(\alpha) &= E(\alpha - E(\alpha))^2 = E(\alpha^2) - E(2\alpha E(\alpha)) + E(E^2(\alpha)) = \\ &= E(\alpha^2) - 2E(\alpha)E(E(\alpha)) + E^2(\alpha) = E(\alpha^2) - E^2(\alpha) \end{aligned}$$

Свойства (Дисперсии)

1. Дисперсия неотрицательна. Если $\Pr(u) = 1$, то $D(\alpha) = 0$
2. $\alpha = \beta + c, \quad c = const, \quad \Rightarrow D(\alpha) = D(\beta)$
3. $\alpha = c\beta, \quad c = const, \quad \Rightarrow D(\alpha) = c^2 D(\beta)$
4. α и β нез. с.в. $\Rightarrow D(\alpha + \beta) = D(\alpha) + D(\beta)$

Док-во

$$2) \quad \alpha = \beta + c$$

$$\begin{aligned} E(\alpha) &= \sum_{u \in U} \alpha(u) \Pr(u) = \sum_{u \in U} (\beta(u) + c) \Pr(u) = \sum_{u \in U} \beta(u) \Pr(u) + \sum_{u \in U} c \Pr(u) = \\ &= E(\beta) + c \end{aligned}$$

$$D(\alpha) = E(\alpha - E(\alpha))^2 = E(\beta + c - E(\beta) - c)^2 = D(\beta)$$

$$3) \quad D(\alpha) = E(c\beta - E(c\beta))^2 = E(c\beta - cE(\beta))^2 = c^2 E(\beta - E(\beta))^2 = c^2 D(\beta)$$

$$4) \quad D(\alpha + \beta) = E(\alpha + \beta - E(\alpha + \beta))^2 =$$

$$\begin{aligned} &= E(\alpha - E(\alpha) + \beta - E(\beta))^2 = E(\alpha - E(\alpha))^2 + 2E((\alpha - E(\alpha)(\beta - E(\beta)))) + \\ &+ E(\beta - E(\beta))^2 = D(\alpha) + D(\beta) + 2(E(\alpha\beta - \alpha E(\beta) - \beta E(\alpha) + E(\alpha)E(\beta))) = \\ &= D(\alpha) + D(\beta) + 2(E(\alpha\beta) - E(\alpha)E(\beta) - E(\beta)E(\alpha) + E(\alpha)E(\beta)) = D(\alpha) + D(\beta) \end{aligned}$$

16 Схема Бернулли

Задача

Стрелок делает 5 выстрелов, какова вероятность того, что он попадет не меньше 4 раз?

p - вероятность попасть в мишень

$$P_5(A) = P_5(4) + P_5(5) = P_5(4) + p^5$$

$$P_5(A) = C_5^4 \cdot p^4(1-p) + p^5$$

hint: мы выбираем, когда стрелок промахнется из всех выстрелов

Опр

$$P_n(m) = C_n^m p^m (1-p)^{n-m} \quad \text{формула Бернулли}$$

n - число попыток m - удачных событий

17 Случайные числа. Схема Уолкера

Замечание

Зачем все это нужно? Мы хотим провести серию неких экспериментов, для этого мы можем использовать метод статистического моделирование. На компьютере можно имитировать случайные эксперименты. В качестве источника случайности мы можем взять генератор случайных чисел (при каждом обращении генератор дает нам какое-то число). В Романовском написано, что эти величины имеют равномерное распределение, но это зависит только от того, какой генератор взять (Если коротко, то равномерное распределение - это, когда у нас нет "перекосов" в сторону каких-то значений).

Пример

Для вычисления площади $sq(A)$ плоской ограниченной фигуры A можно построить содержащий фигуру прямоугольник R (стороны которого \parallel осям коорд.) Будем бить случайными точками в этот прямоугольник. Отношение числа точек, попавших в A, к общему числу точек будет хорошей оценкой площади.

При равномерном распределении $P(\text{попасть в } A) = sq(A)/sq(R)$

Опр (схема Уолкера)

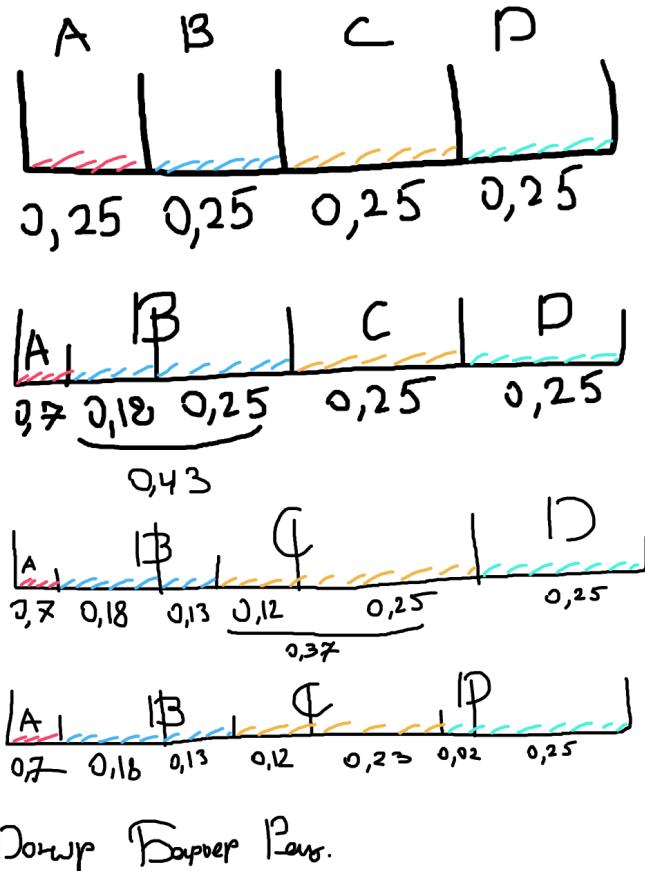
(По-сущи это своеобразный генератор случайных чисел, но с нашим распределением)

Мы хотим создать некую схему, которая сможет нам отвечать, куда мы попали нашей точкой на отрезке $[1, 0]$, какой мы получили исход. Изначально нам даны вероятности этих событий, всего их n. Мы строим таблицу, для каждого исхода запишем $\frac{1}{n}$. То есть, мы изначально предполагаем, что у нас равномерное распределение. Потом мы начинаем это корректировать. Берем событие, которое получило "вероятностной массы" больше, чем остальные. Назовем его донором. Возьмем событие, у которого вероятность ниже, чем мы хотели изначально. Назовем его рецепциентом. Отрежем от донора и отдадим рецепциенту. (То есть, когда мы будем бить точками в этот отрезанный кусок, то он будет относиться уже к рецепциенту, и наш генератор вернет нам другое число). В таблицу нужно еще записать барьеры, которые помогут нам определять, куда попала наша точка. Барьер = остаток донора в $\frac{1}{n}$ отрезка $\cdot 4$. (то есть, если точка правее, то она попала в рецепциента, а иначе в донора) Если у рецепциента стало слишком много массы, то он сам станет донором для другого события. (Резать нужно от его исходного куска в $\frac{1}{n}$ - ую). После того, как все исходы получили нужную вероятность, нам остается научиться быстро определять, куда попала наша точка $x \in [0, 1]$. Мы

можем быстро понять, в какую $\frac{1}{n}$ -ую попала значение. Умножим x на n и возьмем целую часть. Дальше мы смотрим на барьер, если значение x больше барьера, то выбираем реципиента, иначе донора.

Пример

$$P_a = 0,07 \quad P_b = 0,31 \quad P_c = 0,35 \quad P_d = 0,27$$



Родыр Барьер Раст.

$$A \quad 0,7 \cdot 4 \quad B$$

$$B \quad 0,13 \cdot 4 \quad C$$

$$C \quad 0,23 \cdot 4 \quad D$$

$$D \quad 0,25 \cdot 4 \quad D \leftarrow \text{Костыль}$$

18 Двоичный поиск и неравенство Крафта

19 Энтропия. 2 леммы

20 Теорема об энтропии

21 Операции над строками переменной длины

22 Поиск образца в строке (Карпа-Рабина, Бойера-Мура)

23 Суффиксное дерево

24 Задача о максимальном совпадении двух строк

25 Код Шеннона-Фано. Алгоритм Хаффмена. 3 леммы

26 Сжатие информации по методу Зива-Лемпеля

Опр (Алгоритм)

X - Входная фраза (некая строка, которую мы строим)

Точкой обозначена конкатенация

1. Занести все возможные символы в словарь. (им всем будет присвоен код)
2. Считаем один символ из сообщения и добавим его в X
3. Считаем символ Y, если это символ конца сообщения, то вернем код X (он уже лежит в словаре), иначе:
 - (a) Если X.Y уже есть в словаре, то присвоим X = X.Y, перейдем к шагу 3
 - (b) Иначе вернем код для X, добавим X.Y в словарь и присвоим входной фразе значение Y X = Y, перейдем к шагу 3

Пример

Пример плохой из-за тупости Григорьевой. На самом деле, если не добавить все возможные символы в словарь, то декодировать строку будет невозможно, либо нам придется передать вместе со строкой еще и словарь, состоящий из односимвольных фраз

abrakadabra

a - 1 b - 2 r - 3 k - 4 d - 5

ab - 6 br - 7 ra - 8 ak - 9 ka - 10

ad - 11 ab - 12 bra - 13

Output: $\begin{matrix} 1 & 2 & 3 & 1 & 4 & 1 & 5 & 1 & 7 & 1 \\ a & b & r & & & & & & & \end{matrix}$

Опр (Декодирование)

1. Занести все возможные символы в словарь.
2. В X считать первый код сообщения
3. Считать очередной код Y из сообщения, если Y - конец сообщения, то выдать символ, соответствующий коду X, иначе:
 - (a) если фразы под кодом X.Y нет в словаре, то вывести фразу, соответствующую коду X, а фразу с кодом X.Y занести в словарь (! но присвоить ей другой код, а именно: кол-во элементов в словаре + 1)
 - (b) Иначе присвоить фразе код X.Y и перейти к шагу 3

27 Метод Барроуза-Уилера

Опр (Алгоритм)

1. Составляется таблица всех циклических сдвигов строки.
2. Производится лексикографическая сортировка строк таблицы.
3. В качестве выходной строки выбирается последний столбец таблицы преобразования и номер строки, совпадающей с исходной.

Пример

Вход	ц. сдвиги	сортировка	Выход
	<u>abacaba</u>	aabacab	
	bacabaa	abaabac	
	acabaab	<u>abacaba</u>	
abacaba	cabaaba	acabaab	bcabaaa, 3
	abaabac	baabaca	
	baabaca	bacabaa	
	aabacab	cabaaba	

$$\text{BWT}(\text{"abacaba"}) = (\text{"bcabaaa } 3)$$

Опр (Алгоритм обратного преобразования)

Пусть нам дали $\text{BWT}(S) = (A, x)$

Тогда выпишем в столбик A, отсортируем, слева допишем A, снова отсортируем, так n раз, где n - длина строки A. После последней сортировки мы получим, что строка с номером x - S

28 Избыточное кодирование. Коды Хэмминга

29 Шифрование с открытым ключом

30 Сортировки (5 методов)

31 Информационный поиск и организация информации

32 Хеширование

33 АВЛ-деревья

Опр

АВЛ-дерево - сбалансированное по высоте двоичное дерево поиска, для каждой его вершины высота ее двух поддеревьев различается не более чем на 1

Теорема

АВЛ-дерево с n ключами имеет высоту

$$h = O(\log N)$$

Опр

Баланс вершины - разница между высотами ее поддеревьев

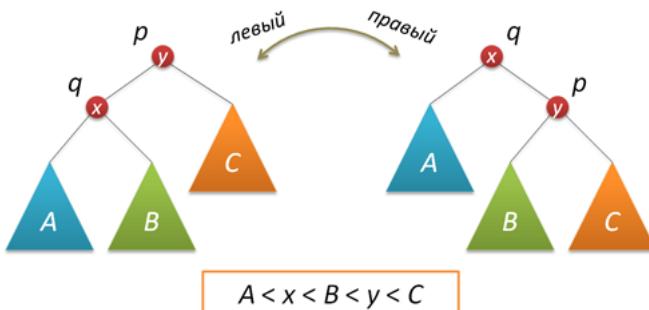
Опр (Балансировка)

Балансировка - операция, которая в случае разницы высот левого и правого поддеревьев $|h(L) - h(R)| = 2$, изменяет связи предок-потомок в поддереве данной вершины так, что разница становится ≤ 1 , иначе ничего не меняет.

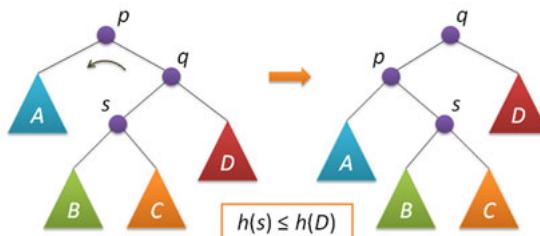
Восстановить баланс можно с помощью поворотов (всего их 4: левый простой, правый простой, левый большой, правый большой)

Простой поворот выполняется при условии $h(s) \leq h(D)$

Простой поворот вправо

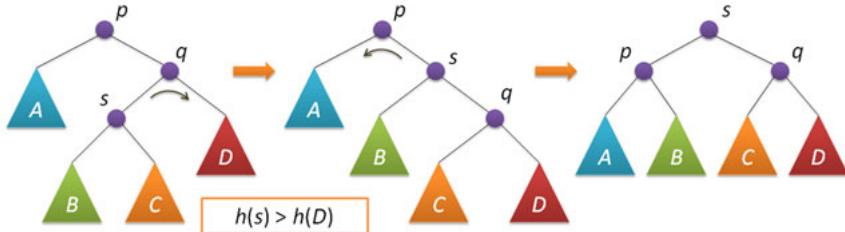


Применение простого поворота



Большой поворот выполняется при условии $h(s) > h(D)$ и сводится к двум простым поворотам

Большой поворот



hint: первым простым поворотом мы отменяем это условие

Опр (Добавление вершины)

Добавляем вершину как в бинарном дереве. Спускаемся вниз, как при поиске ключа t . Если мы стоим в вершине a и нам надо идти в поддерево, которого нет, то делаем ключ t листом, а вершину a его корнем. Дальше поднимаемся вверх и пересчитываем баланс у вершин. Если мы поднялись в вершину i из левого под дерева, то баланс i -ой вершины увеличивается на 1, иначе уменьшается.

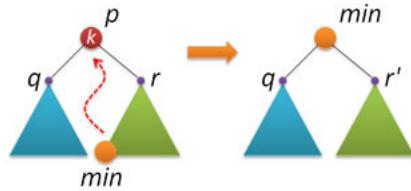
Если мы пришли в вершину и ее баланс = 0 после пересчета, то это значит, что высота под дерева с корнем в этой вершине не изменилась, можно остановить подъем.

Если баланс вершины после пересчета = -2 или 2, то нам необходимо сбалансиовать это поддерево.

Добавление работает за $O(\log n)$, т.к. мы рассмотрим не больше, чем $O(h)$ вершин

Опр (Удаление ключа)

1. найдем удаляемый ключ r в дереве (если не нашли, то ничего не делаем)
2. в правом его поддереве найдем наименьший элемент \min
3. поменяем местами поменяем местами r и \min
4. удалим r
5. сбалансируем все, что выше r



При удалении возможна ситуация, когда вершина r не имеет правого поддерева, тогда по св-ву АЛВ-дерева либо эта вершина имеет слева единственный дочерний узел, либо она является листом. В обоих случаях мы просто удаляем r и возвращаем указатель на левое поддерево.

34 B-деревья

35 Биномиальные кучи

36 Основные определения теории графов

37 Построение транзитивного замыкания

38 Обходы графа в ширину и глубину. Топологическая сортировка

39 Связность. Компоненты связности и сильной связности

40 Алгоритм поиска контура и построение диаграммы порядка

41 Теорема о связном подграфе

42 Деревья. Теорема о шести эквивалентных определениях дерева

43 Задача о кратчайшем остовном дереве. Алгоритм Прима

44 Алгоритм Краскала

45 Задача о кратчайшем пути. Алгоритм Дейкстры

46 Алгоритм Левита

47 Задача о кратчайшем дереве путей

48 Сетевой график и критические пути. Нахождение резервов работ

49 Задача о максимальном паросочетании в графе.
Алгоритм построения

50 Теорема Кенига

51 Алгоритм построения контролирующего множества

52 Задача о назначениях. Венгерский метод

53 Задача коммивояжера. Метод ветвей и границ

54 Метод динамического программирования. Задача линейного раскроя

55 Приближенные методы решения дискретных задач.
Жадные алгоритмы

56 Алгоритмы с гарантированной оценкой точности.
Алгоритм Эйлера

57 Жадные алгоритмы. Задача о системе различных представителей

58 Приближенные методы решения дискретных задач

59 Конечные автоматы

60 Числа Фибоначчи. Производящие функции

61 Числа Каталана

62 ?Алгоритм Кристофидеса (возможно будет)