# Дискретный анализ

## Лекция 2

Наборы из нулей и единиц

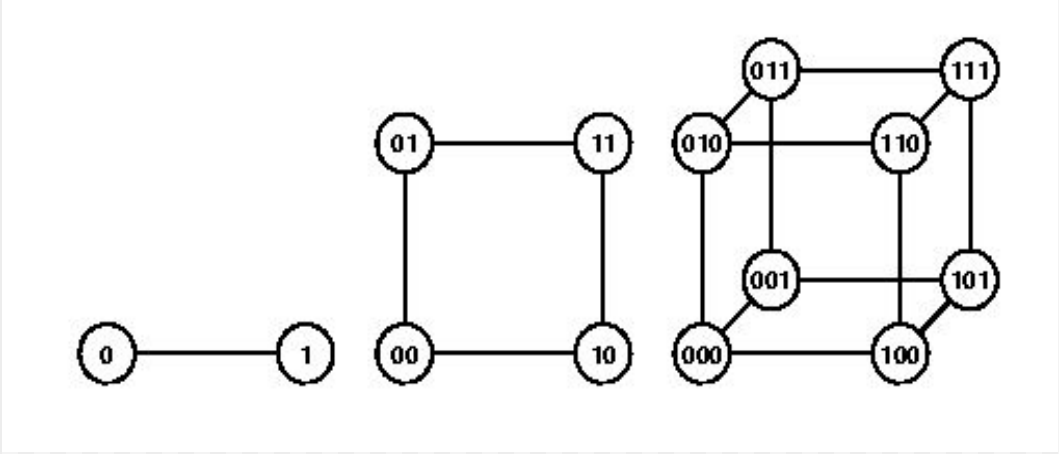
# Трактовки наборов из *k* нулей и единиц

- Вершина куба
- Характеристический вектор множества
- Набор булевых значений
- Картинка
- Двоичное представление целого числа
- Состояние памяти компьютера
- Символ кодировки
- Путь в целочисленной решетке
- Результаты случайных испытаний (например, бросаний монеты)
- И другие, более сложные.

# Вершина куба

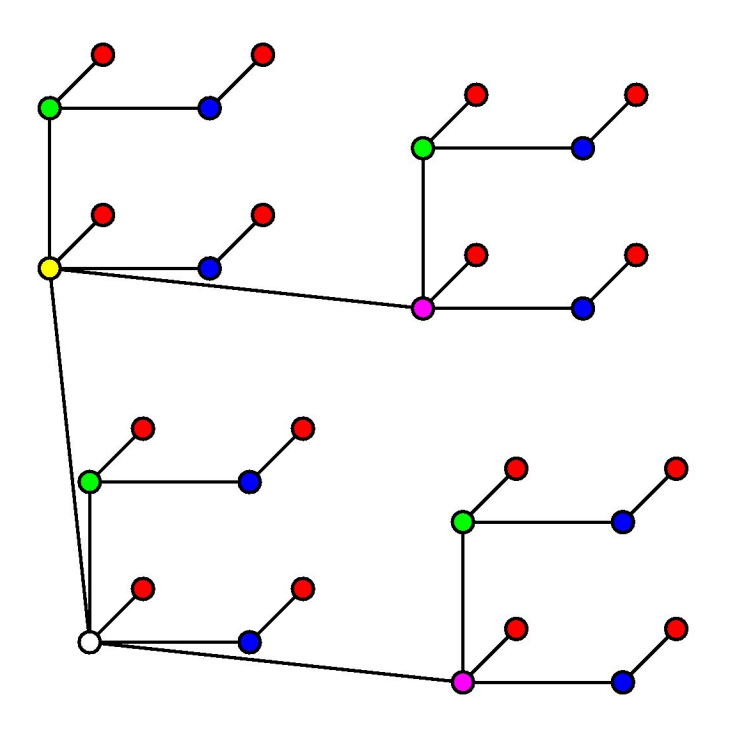
- Начнем с того, что набор из $k$ нулей и единиц можно трактовать как вершину $k$–мерного единичного куба, у которого одна из вершин находится в начале координат, а исходящие из нее ребра лежат на координатных осях.
- Такой куб легко представить себе для $k$, равного 1, 2 и 3, а для больших размерностей все «выглядит аналогично».

# Вершина куба-2




- Вы видите здесь кубы для первых трех размерностей. Попробуйте сами нарисовать куб размерности 4 и 5.
- Отмечу, что такое представление 0-1 наборов оказалось очень полезным.

# Вершина куба-3



- Вот 5-мерный куб, в котором проведены только ребра, необходимые для минимальной связи между вершинами.
- Так устроено соединение процессоров во многопроцессорных компьютерах. Но в них обычно вершин-процессоров больше, типично $k$=16.

# Набор булевых значений

Что такое булевы значения?

Джордж Буль в 1854 году предложил систему исчисления логических высказываний, введя новый тип логических величин, принимающих два значения – ИСТИНА и ЛОЖЬ (True и False).

Для этих (булевых) величин были введены своеобразные операции, вполне естественные с точки зрения формальной логики.

Сейчас булевы величины широко используются в программировании.

Мы сейчас рассмотрим булевы величины и операции с ними.

**George Boole**

# Булевы значения и операции

- Примеры булевых значений.
- “п > 3.1415926” - это ИСТИНА
- “п > 4.011” - это ЛОЖЬ
- “x > y+3” - это может быть ИСТИНОЙ, а может быть ЛОЖЬЮ в зависимости от значений x и y.
- Операция отрицание или НЕТ или NOT или ¬ (а в языке программирования Си - !) имеет один логический операнд (она одноместная) и вырабатывает противоположное ему значение:
- ¬ True = False
- ¬ False = True
- ¬ (п > 3.1415926) = False

# Операция конъюнкция

- Эта операция называется еще логическим И и обозначается AND, а также $\wedge$ или &. У нее два булевских операнда и результат операции также булевское значение «оба операнда истинны».
- Например, $(x>3)\wedge(x<7)$ истинно для всех $x$, больших 3 и меньших 7.

# Операция дизъюнкция

- Эта операция называется логическим ИЛИ и обозначается OR, а также $\vee$ или ||. У нее два булевских операнда и результат операции также булевское значение «хотя бы один из операндов истинен».
- Например, $(x>3)\vee(x<7)$ истинно для всех $x$, больших 3, и для всех, меньших 7.

# Операция эквиваленция

- Эта операция обозначается EQU, а также ≡ . У нее два булевских операнда и результат операции также булевское значение «значения операндов совпадают».
- Например, $(x>3)\equiv(x<7)$ истинно для всех $x$, больших 3, и меньших 7 (вариант, когда оба операнда ложны, здесь невозможен).

# Операция «исключающее ИЛИ»

- Эта операция обозначается XOR (от eXclusive OR), она общепринятого обозначения не имеет, хотя появилось обозначение $\underline{\vee}$. У нее два булевских операнда и результат операции также булевское значение «значения операндов не совпадают».
- $(x>3)$ XOR $(x<7)$ истинно для всех $x$, больших 3, и для всех, меньших 7.
- Эта операция имеет удобное свойство
- $(a \text{ XOR } b) \text{ XOR } b = a$
- которое используется в компьютерной графике.

# Операция импликация

- Редко используемая операция «следования» обозначается IMP (от IMPlication), или $\supset$. У нее два булевских операнда и результат операции также булевское значение «либо ложен первый операнд, либо истинен второй».
- $(x>3)$ IMP $(x<7)$ истинно для всех $x$, меньших 7.

# Таблица логических операций

- Все названные двуместные операции можно свести в таблицу

| x | y | $x \wedge y$ | $x \vee y$ | $x \equiv y$ | $x \underline{\vee} y$ | $x \supset y$ |
|---|---|---|---|---|---|---|
| F | F | F | F | T | F | T |
| F | T | F | T | F | T | F |
| T | F | F | T | F | T | T |
| T | T | T | T | T | F | T |

# 0-1 представление булевых значений

- Естественна трактовка нуля и единицы как логических значений, соответственно, False и True.
- Все логические операции легко переписать для этого представления.
- Но и 0-1 набор можно представлять как набор логических значений и все перечисленные логические операции выполнять над наборами-операндами покомпонентно.

# Пример логических операций над 0-1 наборами

| | |
|---|---|
| x | 0000000111111111 |
| y | 0001111000111111 |
| $x \wedge y$ | 0000000000111111 |
| $x \vee y$ | 0001111111111111 |
| $x \equiv y$ | 1110000000111111 |
| $x \underline{\vee} y$ | 0001111111100000 |
| | 1111111000111111 |

# Логические функции

- Комбинируя значения отдельных компонент логического набора в более сложных операциях, можно составлять более сложные функции от логических аргументов.
- Функция от логических значений, принимающая логические значения, называется логической функцией.
- Заметим, что каждая логическая функция может быть задана таблицей истинности -перечислением тех наборов аргументов, которым соответствует значение True.
- Используя эту таблицу, всегда можно представить логическую функцию в виде дизъюнкции конъюнкций.

# ДНФ (дизъюнктивная нормальная форма)

- Функция $f$ может быть всегда представлена так
- $f(x_1,x_2,\dots,x_k)=\bigvee_{a\in T}\left(\bigwedge_{i\in 1:k} v(x_i,a)\right)$
- где $T$ – таблица истинности. Каждый входящий в нее набор $a$ определяет для каждой переменной $x_i$ способ ее вхождения в соответствующую $a$ дизъюнкцию: входит сама переменная или ее отрицание. Эта зависимость спрятана в функции $v$.
- Интересна задача сокращения ДНФ до минимума.

# Характеристический вектор множества

- Если сопоставить элементы конечного множества $S$ мощности $k$ позициям в наборе из  нулей и единиц, то подмножествам можно сопоставить такие наборы. Пусть для простоты  $S=1:9$. Подмножеству $A=\{1,2,6,7,9\}$ соответствует набор 110001101.
- Такой 0-1 набор $\chi_A$ называется характеристическим вектором множества .
- Операции над множествами легко моделируются логическими операциями над их характеристическими векторами.
- Например,
- $\chi_{A\cap B} = \chi_A \wedge \chi_B$

# Двоичное представление числа

- Вы, конечно, знаете, что каждое натуральное число однозначно представимо в виде суммы степеней 2, причем каждая степень в этой сумме появляется не больше одного раза, т. е. с коэффициентом 0 или 1. Эти коэффициенты составляют представление числа в двоичной системе счисления.
- Например, 83=64+16+2+1=1010011.
- Таким образом, каждый набор из $k$ нулей и единиц соответствует какому-либо числу в диапазоне от 0 до $2^k$-1.

# Степени двойки

- Степени двойки из-за использования двоичной системы встречаются так часто, что некоторые из них  полезно помнить наизусть.
- Число 2^{10} – это наша тысяча. Оно обозначается K и называется кило, (ср. килобайт).
- 2^{20}= 1048576 – это миллион (мега). Дальше следуют гига и тера, найдите их значения сами.

| k | 2^k |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |
| 6 | 64 |
| 7 | 128 |
| 8 | 256 |
| 9 | 512 |
| 10 | 1024 |

# Арифметические действия с двоичными числами

- Вы, конечно, знаете, как удобно выполнять арифметические действия с двоичными числами.
- Напомним, начиная с самого простого, с прибавления единицы.
- Как прибавить единицу к числу 1000111010001111 ?
- Ответ: двигаясь от конца к началу, заменять единицы на нули, а встретив нуль, заменить его на единицу и остановиться.
- В рассматриваемом случае получится 1000111010010000 . Красным выделена изменившаяся часть.

# Арифметические действия с двоичными числами-2

- Разработано много эффективных схем сложения, вычитания, умножения и деления двоичных чисел. Мы ими заниматься не будем.
- Но нужно упомянуть об особом случае умножения и деления двоичного числа на степень двойки $2^k$. Умножение выполняется приписыванием к записи числа $k$ нулей. В компьютере это соответствует сдвигу записи числа на $k$ позиций влево, и существуют машинные команды сдвига.
- Мы говорим влево, считая, что число записано в компьютере так, как мы привыкли писать на бумаге – младшие разряды правее левых. В некоторых случаях полезно считать, что запись идет в противоположном направлении. К этому вопросу мы еще вернемся.
- А про команды сдвига нужно поговорить еще.

# Арифметические действия с двоичными числами-3

- Ясно, что сдвиге влево старшие разряды теряются (а при делении – сдвиг происходит вправо, и теряются младшие разряды).
- Существуют команды циклического сдвига, при которых вытесняемые разряды занимают освобождающиеся места с другого конца записи. Например, циклический сдвиг на три разряда выглядит так
- 12345678 → 45678123
- Попробуйте написать программу циклического сдвига элементов массива, не использующую большой дополнительной памяти (порядка длины массива).

# Более удобные системы счисления

- Двоичная система счисления удобна для компьютера, но неудобна для человека – слишком длинные получаются записи чисел:
- 01010111000101110101101 0
- Даже длину такой записи трудно определить!
- Компромиссом между интересами человека и машины являются системы счисления с основаниями, близкими к 10, но являющимися степенью двойки – 8 и 16.

# Восьмеричная система счисления

- В этой системе 8 цифр – 0, 1, 2, 3, 4, 5, 6, 7. Каждому разряду восьмеричной системы соответствуют 3 разряда двоичной системы, и переход очень прост:
- 0 – 000  2 – 010  4 – 100  6 – 110
- 1 – 001  3 – 011  5 – 101  7 – 111
- Число с предыдущего слайда в этой системе записывается так:
- 010 101 110 001 011 101 011 010
- 2  5  6  1  3  5  3  2
- Итак, 25613532.
- Восьмеричная запись сейчас используется относительно редко.

# Шестнадцатеричная система счисления

- В этой системе 16 цифр. Десять обычных – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Еще шесть – это буквы A, B, C, D, E, F. Каждому разряду соответствуют 4 разряда двоичной системы:
- 0 – 0000  4 – 0100  8 – 1000  C – 1100
- 1 – 0001  5 – 0101  9 – 1001  D – 1101
- 2 – 0010  6 – 0110  A – 1010  E – 1110
- 3 – 0011  7 – 0111  B – 1011  F – 1111
- Число из примера в этой системе записывается так:
- 0101 0111 0001 0111 0101 1010
- 5  7  1  7  5  A
- Итак, 57175A.
- 16-ричная запись используется очень широко. Особенно при обсуждении вопросов памяти компьютера.

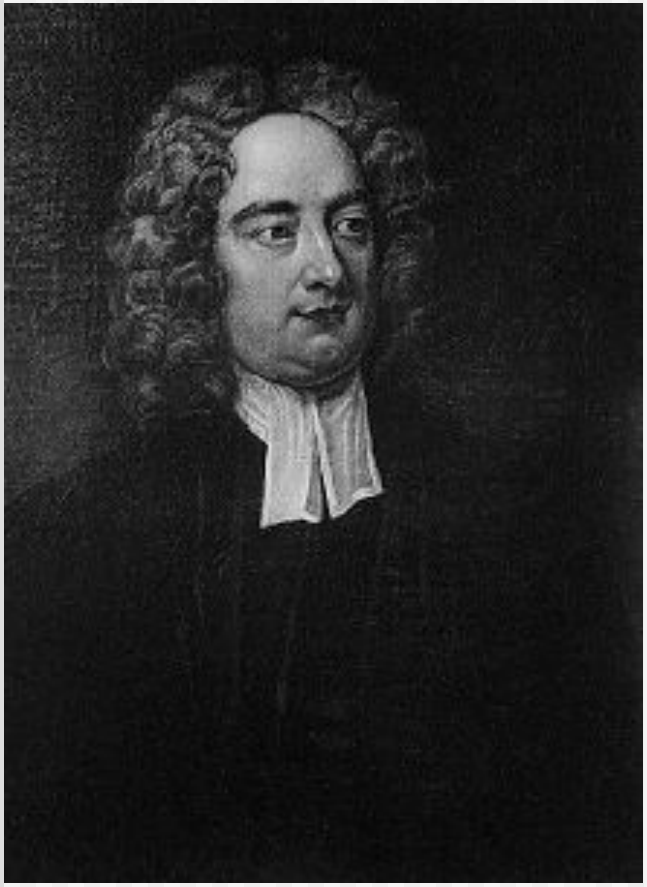
# Состояние памяти компьютера

- Вся память компьютера состоит из мельчайших магнитных элементов, каждый из которых может находиться в одном из двух состояний. Одно из них связывается с нулем, а другое с единицей. Машина может «прочесть» состояние магнита, а значит и хранимую им двоичную цифру. Такая цифра называется битом (bit).
- Так как этих цифр очень много, то с ними невозможно работать без укрупнения. Первое укрупнение – то объединение битов в восьмерки – байты (byte). Байт – минимальная адресуемая единица памяти. Это значит, что машинная команда, работающая с памятью может добраться только до всего байта целиком или до еще более крупной единицы памяти.

# Машинное слово

- Пара подряд идущих байтов называется машинным словом (word). Слово располагается так, чтобы минимальный из адресов байтов был четным (это называется выравниванием – слово выравнивается на четный адрес).
- Один из байтов старший, а другой младший.
- Где расположен младший байт? В разных компьютерах по-разному. Если впереди, то говорят, что компьютер мелкоконечный (little endian – этот термин ввел Джонатан Свифт в «Путешествии Гулливера»). Если сзади, то компьютер крупноконечный (bigendian).
- Наши персональные компьютеры – мелкоконечные, и когда вы находите в памяти целое число, занимающее два байта, например, $515 = 0203_{16}$, то в первом байте пишется 03, а во втором – 02.

# «Путешествия Гулливера»



- Процитируем:
- It is computed, that 11 000 persons have, at several times, suffered death, rather than submit to break their eggs at the smaller end. Many hundred large volumes have been published upon this controversy: but the books of the Big-Endians have been long forbidden, and the whole party rendered incapable by law of holding employments.
- *Jonathan Swift,*
- *The Travels of Gulliver*

# Побайтовое кодирование символов

- Важным событием в развитии информатики (Computer Science) и вычислительной техники было принятие байта в качестве единицы при кодировании символов. Вначале такие коды были приняты фирмой IBM при разработке знаменитого комплекса IBM-360, а сейчас превратились в общемировой стандарт.
- В Советском Союзе внедрение машин Единой Серии (ЕС), клонировавших IBM-360, байты появились «автоматически». Было очень удобно, что в $256=2^8$ кодовых возможностей свободно умещаются и заглавные и строчные буквы и латиницы и кириллицы.

# Таблица ASCII

- Сейчас самым важным стандартом (их несколько) является ASCII.
- Это аббревиатура для American Standard Code for Information Interchange.
- Он стал мировым стандартом кодировки для практически всех компьютеров. В этом коде на каждый символ приходится по одному байту, причем стандарт фиксирует верхнюю половину таблицы с кодами от 0 до 127.
- Представив байт двумя 16-ричными цифрами, мы можем сказать, что это коды от 00 до 7F. Некоторые сведения об этой таблице хорошо запомнить.

# Таблица ASCII (продолжение)

- Первые две строчки заполнены служебными символами (типа возврата каретки, перевода строки, табуляции, звонка, конца передачи).
- Следующая строка начинается с 20 – кода пробела. Дальше идут различные специальные знаки, которые можно не помнить.
- Дальше идет строка цифр: от 30 для 0 до 39 для 9. Остаток заполнен спецзнаками.
- Строки 4 и 5 заполнены прописными латинскими буквами: 40 для @ (at «коммерческое», а совсем не собака), 41 для A и т.д. вплоть до  5A для Z.
- Строки 6 и 7 аналогично заполнены строчными латинскими буквами. 60 используется для ` (обратного апострофа).

# Таблица ASCII (продолжение 2)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | |
| 2 | ␣ | | | | | | | | | | | | | | | |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | | | | | |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | | | | | |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | | | | | |

# Таблица ASCII (продолжение 3)

- Вторая половина таблицы ASCII существует в нескольких вариантах (они называются codepages – кодовые страницы).
- Для нас наиболее важны следующие страницы
- 437 – MS DOS Extended
- 1252 – Windows
- 866 – DOS Cyrillic
- 1251 – Win Cyrillic

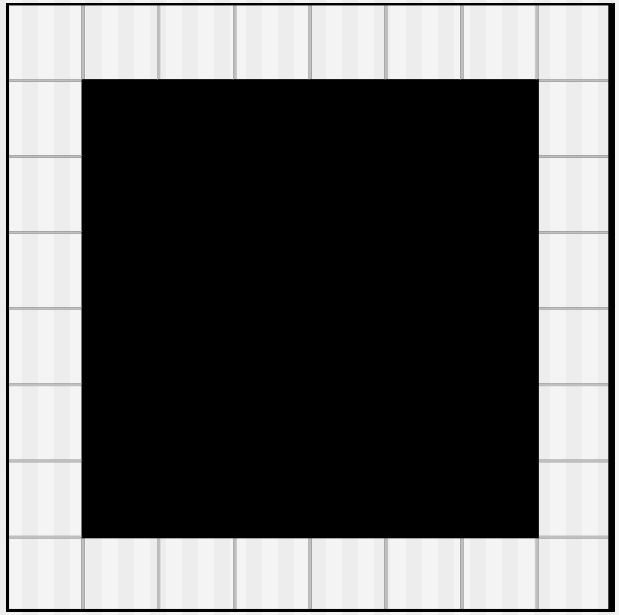
# UNICODE и родственные кодировки

- Сейчас идет активный перевод программного обеспечения на двухбайтовую систему кодировки. Принят международный стандарт, именуемый UNICODE. Этот стандарт имеет «кодовое пространство» в $2^{16}$=65536 символов, чего вполне достаточно для всех языков мира, включая полный набор китайских иероглифов (сейчас в китайских компьютерных системах используется сокращенный набор в примерно 5000 знаков).
- Для более удобного перехода ASCII↔UNICODE разработана специальная кодировка UTF-8 (Unicode Transport Format), о которой речь пойдет в дальнейшем.

# Картинка

- Двухцветную (скажем, черно-белую) картинку можно свести к некоторому «растру» - прямоугольной решетке и затем рассматривать как подмножество множества точек этой решетки. А для задания множества можно построить его характеристический вектор («вытянув» предварительно решетку в линию, т.е. линейно пронумеровав точки решетки).
- Например, следуя Малевичу, нарисуем «черный квадрат» 6×6 на поле 8×8.

# Черный квадрат



Черный квадрат и его представление в одну строку. Конечно, вместо белых квадратиков нужно писать нули, а вместо черных – единицы.
Есть и более экономные представления картинок

00000000 01111110 01111110 01111110 01111110 01111110 01111110 00000000

# Передача по двоичному каналу связи

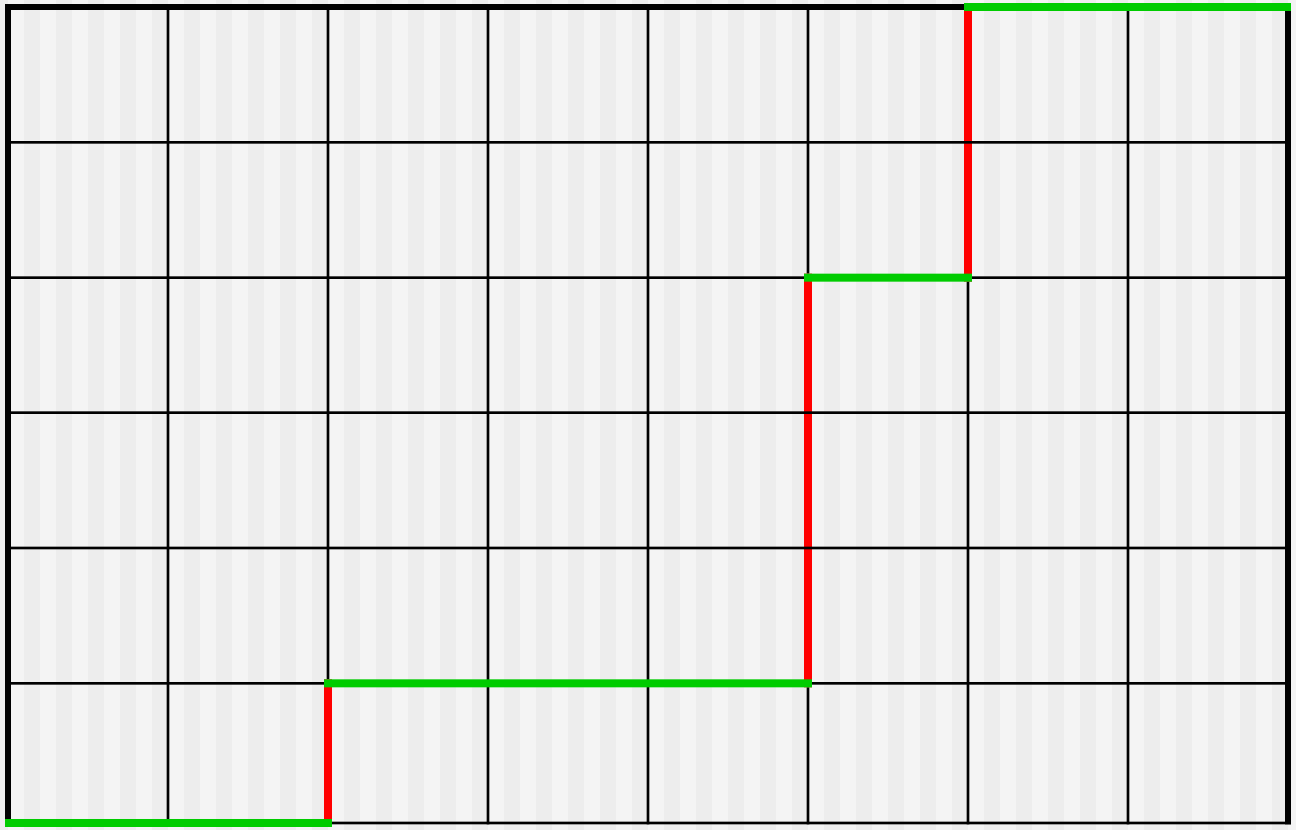
- В технике связи передачу по каналу связи очень часто рассматривают как последовательность импульсов, находящихся на двух уровнях (сигнал и нет сигнала). Таким образом, информация, передаваемая по каналу связи, рассматривается как 0-1 последовательность.
- Смысл нулей и единиц в этой последовательности может быть очень разнообразным и хитрым. Некоторые варианты мы в дальнейшем рассмотрим (сжатые тексты и изображения, компьютерные программы, запись звука, шифрограммы).
- Но чаще всего – это тексты, о байтовом представлении которых уже говорилось.

# Результаты случайных испытаний

- В теории вероятностей принято рассматривать случайные последовательности. Для примера их считают результатами бросания монеты: при падении монета может упасть на разные стороны, которые традиционно называются Гербом и Решеткой (Решкой). Вот и получается
- ГГГРРГРГГРРРРГРГРГГГРГ
- - те же нули и единицы.

# Путь в целочисленной решетке

0-1 набор иногда полезно представлять путем на прямоугольной решетке. Вот путь для набора 11011100010011. По разности координат начала и конца легко судить о числе нулей и единиц в наборе.



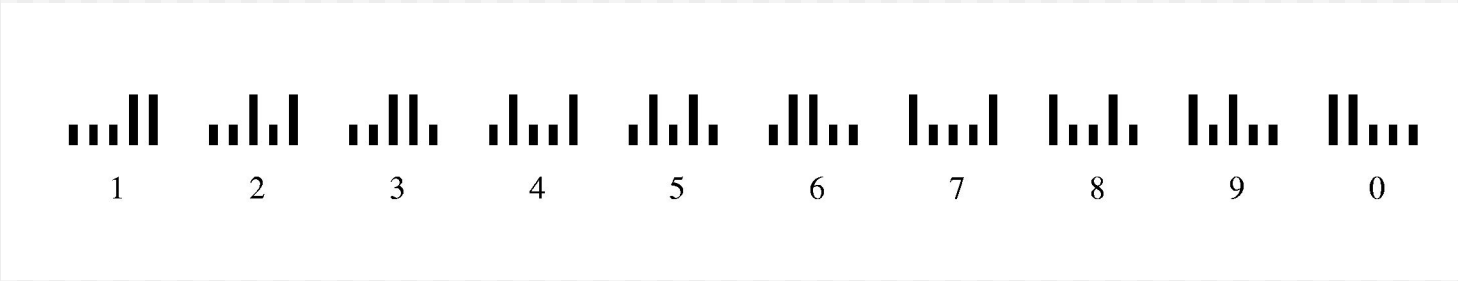
# Штрихкоды

- Нули и единицы, несущие информацию, могут иногда представляться очень своеобразно. Например, вы уже встречались, вероятно, с полосками на упаковках товаров. Они называются штрихкодами (barcodes) и предназначены для быстрого считывания информации специальными устройствами.
- Сейчас есть очень много вариантов штрихкодов, специально приспособленных для тех или иных ситуаций. Рассмотрим только некоторые из них.

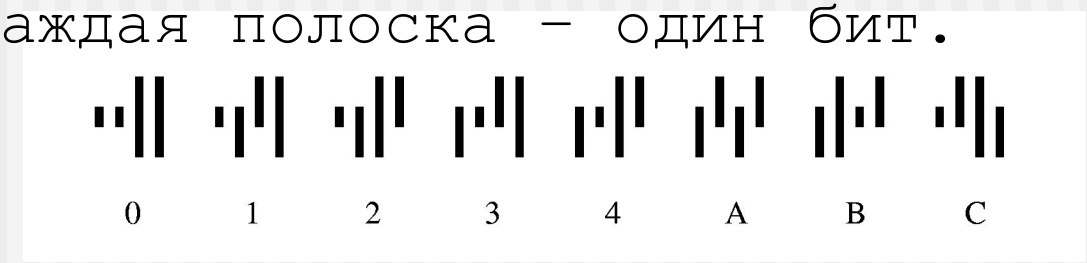
# Штрихкоды-1

Ручной сканер считывает штрихкод с упаковки сока. (фото С.Е.Столяра)

# Штрихкоды-2

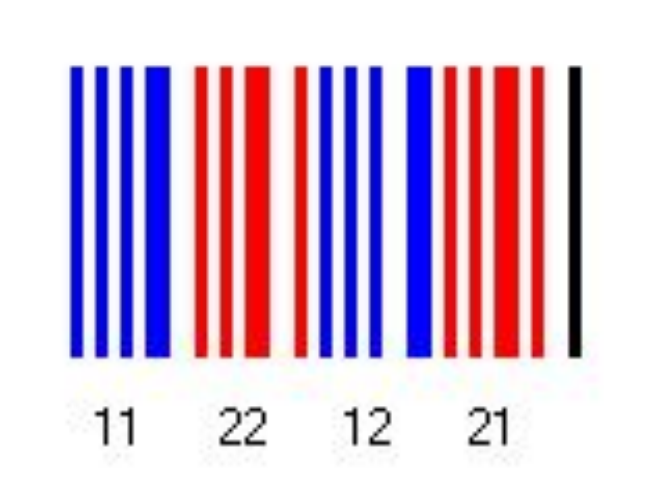



Почтовые штрихкоды США. Код «2 из 5»,
каждая полоска – один бит.




Почтовые штрихкоды Великобритании. Каждая
полоска – два бита.

# Штрихкоды-3




**1 – 0001**
**2 - 0010**

Перемежающиеся коды. Полоски двух символов идут через одну (каждый символ одного цвета). Бит кодируется шириной полоски. Завершающая полоска в конце нужна для определения ширины последней белой полоски.

# Представление чисел с плавающей точкой

- Числа с плавающей точкой представляются в компьютере приблизительно. Сейчас широко распространен стандарт IEEE, в котором фиксированы две формы числа – 32-битовая и 64-битовая.
- Рассмотрим первую из них. Число занимает двойное слово, и составляющие его 32 бита $b_1,\ldots,b_{32}$ разбиты на три поля (вот как: 0-1 набор разбивается на поля, и только это разбиение нам сейчас важно):
- $s = b_1$ - знак числа
- $e = b_{2:9}$ - порядок числа
- $f = b_{10:32}$ – дробная часть нормализованной мантиссы
- Таким образом,
- $x = (-1)^s \times 2^{e-127} \times (1.f)$

# Перебор 0-1 наборов

- Сейчас мы ответим на три вопроса:

1. Сколько элементов в множестве $B^k$ ?
2. Как эти элементы перенумеровать ?
3. Как эти элементы перебрать ?

- Ответы на первые два вопроса мы уже дали раньше: $|B^k|=2^k$, и каждый набор можно рассматривать как двоичное представление целого числа, которое и является его номером. Например, $\#(0101)=5$.

# Перебор 0-1 наборов - 2

- Как же эти элементы перебрать ?
- Очень просто: начать с нулевого набора, которому соответствует число 0, а далее прибавлять о единице (работая прямо с двоичным набором, мы это уже умеем).
- 00000 01000 10000 11000
- 00001 01001 10001 11001
- 00010 01010 10010 11010
- 00011 01011 10011 11011
- 00100 01100 10100 11100
- 00101 01101 10101 11101
- 00110 01110 10110 11110
- 00111 01111 10111 11111

# Перебор 0-1 наборов – 3

- Однако в некоторых случаях такая форма перебора неудобна из-за того, что при переходе от одного набора к другому текущий набор сильно изменяется.
- Вопрос: нельзя ли осуществить перебор так, чтобы при каждом переходе изменение было только в одном бите.
- Ответ: это возможно.
- Мы покажем эту возможность дважды:
- Как математики и
- Как программисты.

# Перебор 0-1 наборов – 4М

- Как математики:
- 0 0 0
- 0 0 1
- 0 1 1
- 0 1 0
- 1 1 0
- 1 1 1
- 1 0 1
- 1 0 0

В первой колонке запишем в половине строк нули, а дальше единицы. Заполним первую половину наборами длины k-1, а затем зеркально отобразим эти наборы во второй половине. А наборы меньшей длины будем перебирать так же.

# Перебор 0-1 наборов – 4П

Как программисты:

**Запишем перебор наборов по возрастанию и с единичными «мутациями»:**

| | | | | | |
|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** | **0** |
| **1** | **0** | **0** | **1** | **1** | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** | **0** | **0** |

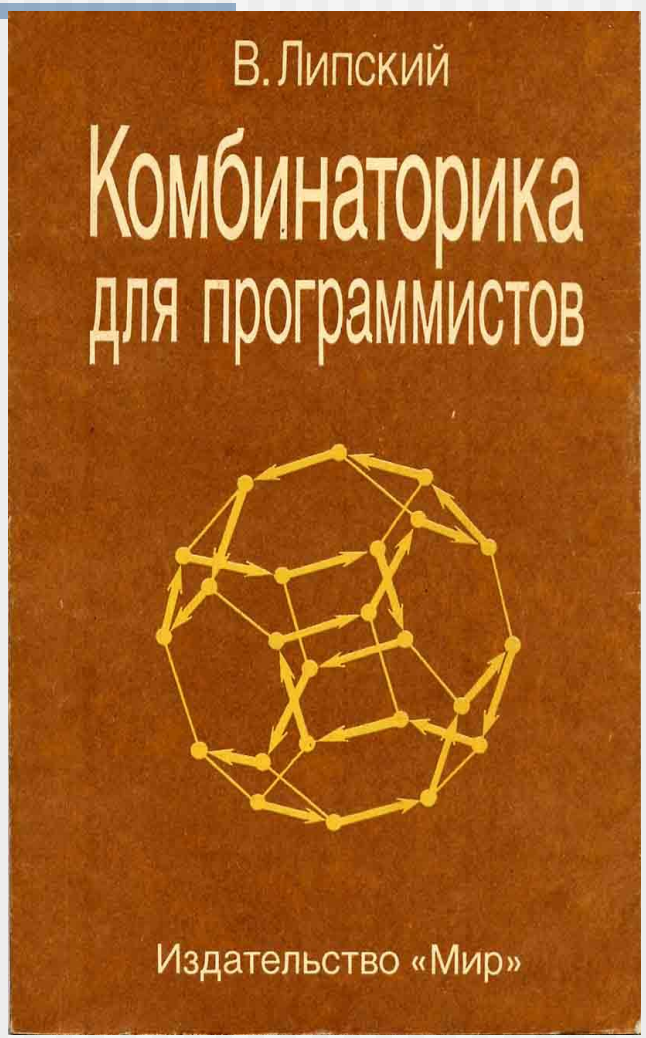
- Видите? Там, где в правой таблице меняется значение бита, в левой появляется 1. (докажите)
- Значит, нужно иметь два рабочих 0-1 набора. В первом моделировать прибавление 1, а во втором, зная позицию изменения, менять значение бита.
- Эта последовательность наборов называется кодом Грея.

# Дополнительная литература-1

- Это действительно классический западный учебник по основным алгоритмам информатики. Книга дорогая, но очень полезная.
- ISBN 5-900916-37-5

# Дополнительная литература-2

- Книг по комбинаторике очень много, но по программистским вопросам только одна.
- Она была издана издательством «Мир» в 1988 году тиражом 45000 экземпляров, так что иногда встречается.



# Экзаменационные вопросы

3. Логические значения и операции
4. Логические функции и ДНФ
5. Характеристический вектор множества
6. Кодировка ASCII и ее варианты
7. UNICODE и UTF-8
8. Штрихкоды (по книжке)
9. Перебор двоичных наборов и код Грея