Три непростые игрушки

или как решать сложные задачи информатики, если на знать, что они сложные … и из информатики

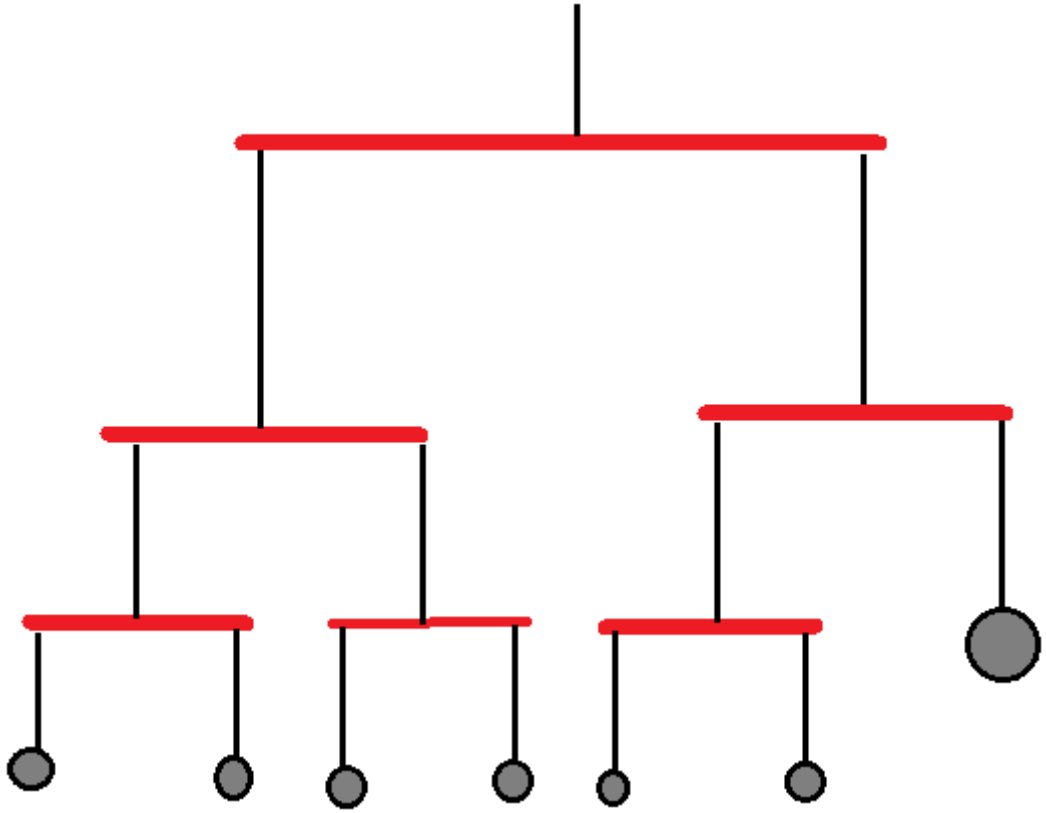
Станислав Протасов

старший научный сотрудник университета Иннополис

Гирьки

Хочу вот такую игрушку!

Все грузики уравновешены на середине палочки



Игрушку привезли …
в разобранном состоянии :(

- Грузики весом **[1]**x2, **[2]**x1, **[4]**x3, **[16]**x1, **[32]**x1

- Как собирать-то будем?

Сжимающее префиксное кодирование!

Кодирование

- Представление символов одного алфавита (например, букв) символами другого алфавита (например, двоичных символов)
  - Можно кодировать 1-1, 1-много, много-1, много-много

- ASCII(Z) = **90** = 1011010_2
- MORSE(Z) = **– – • •**
- PARITY(001101_2) = 0011011_2

Код переменной длины

- Кодирование постоянной длины – Unicode (16), ASCII (8)
- Кодирование переменной длины – UTF-8 (8-16)
- Префиксное кодирование – ни одно слово кодирующего алфавита не является началом другого

Пример префиксного кода переменной длины
A = 0, B = 10, C = 11

Пример

- А – 1000
- Б – 1001
- В – 00
- Ж – 01
- Х – 101
- У – 11

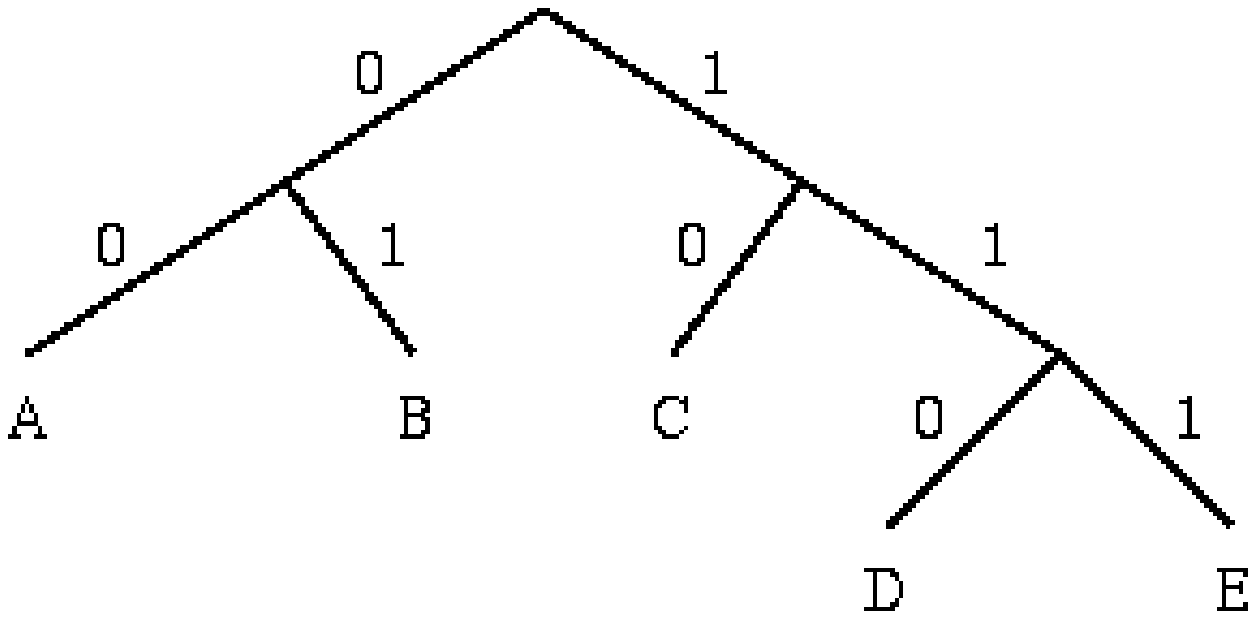
000101010111101 - ?

Почему сжимающий?

- Для кодирования алфавита в 6 символов кодом **фиксированной** длины нужно $\lceil log_2(6) \rceil = 3$ бита
- А - 000
- Б - 001
- В - 010
- Ж - 011
- Х - 100
- У - 101

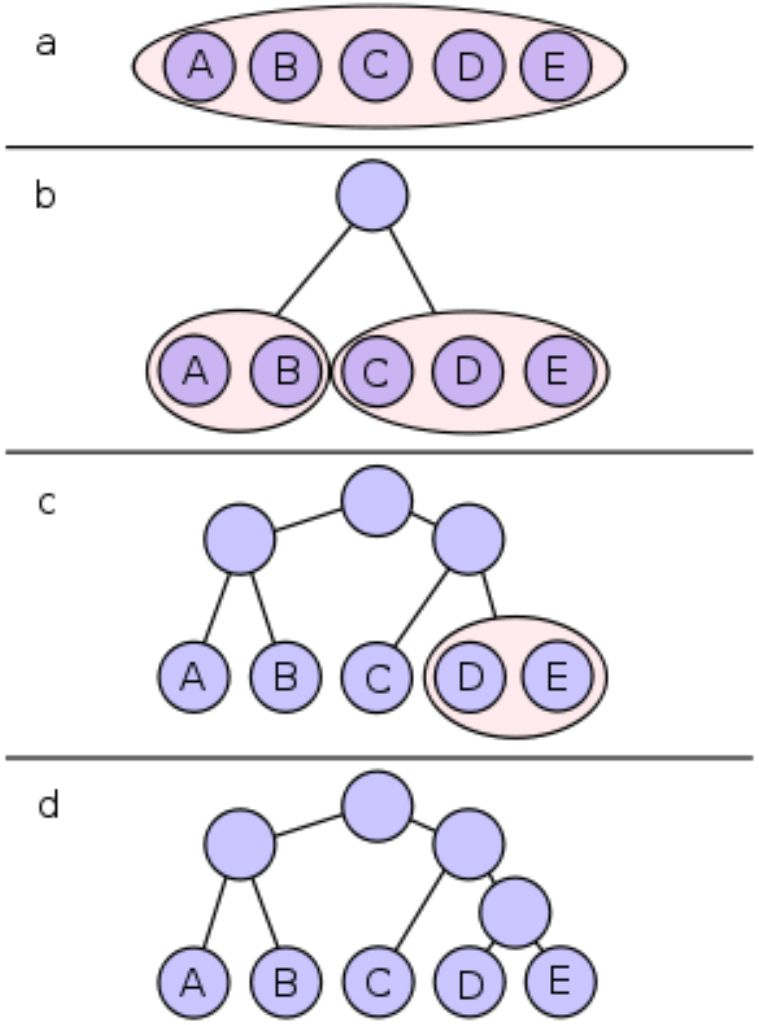
010011011011101100 – 18 бит

Как кодировать?

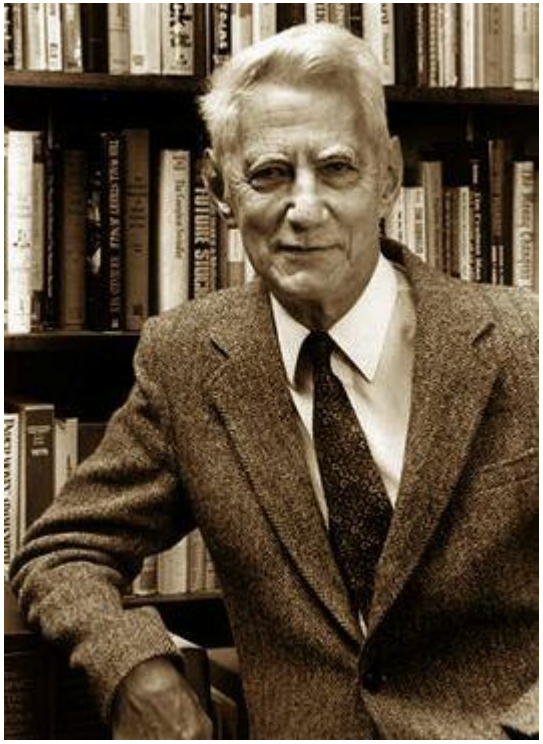


Собирать сверху вниз – код Шеннона-Фано (1948)

- «Вес» буквы – частота её повторений в тексте
- Делим буквы на 2 группы максимально близких по сумме «весов»
- Повторяем, пока в каждой группе не останется по одному символу (все буквы оказались в листьях дерева)

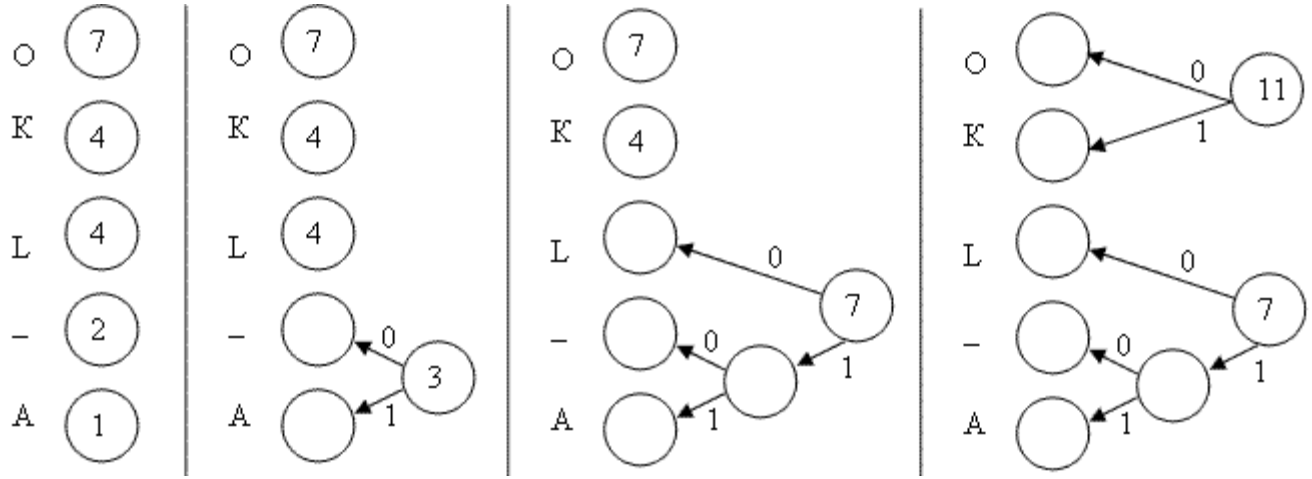


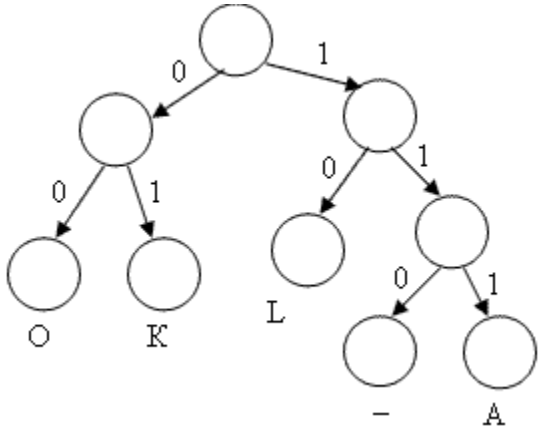
Клод Шэннон



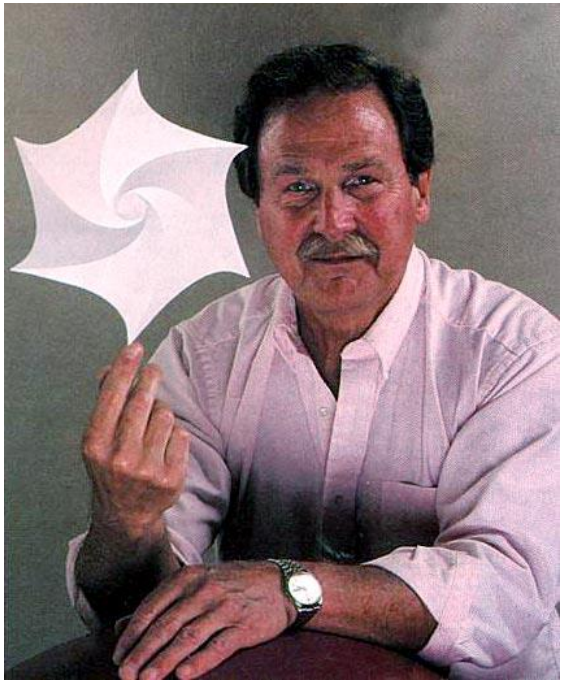
- Отец теории информации, математик. Ввёл понятие **бит**
- Параллельно с Котельниковым предложил теорему о дискретном кодировании аналоговых сигналов. Благодаря этой теореме у нас есть цифровые форматы хранения музыки (началось с CD) и сотовая связь

Собирать снизу вверх – код Хаффмана (1952)





Дэвид Хаффмана



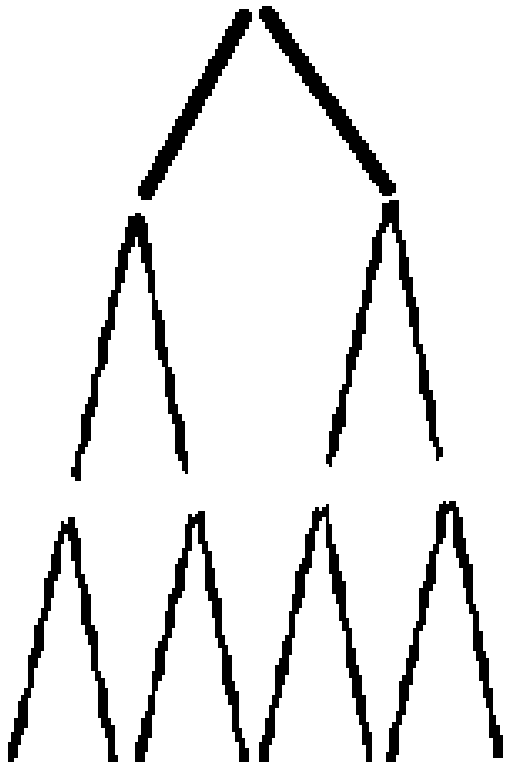
- Первопроходец в сфере теории информации
- Внёс вклад в математическое оригами и электронику
- Награждён бесчисленными наградами за создание **минимально-избыточных кодов**
  - Используется в MP3, ZIP (deflate), JPEG
  - Наследники – арифметическое кодирование, LZW (GIF)

И что дальше?

- Июнь 2016 – RAISR – алгоритм восстановления деталей на изображениях
- Март 2017 – Google опубликовала алгоритм Guetzli, который является улучшением JPEG, позволяет сжимать на 35% лучше

Карточный домик

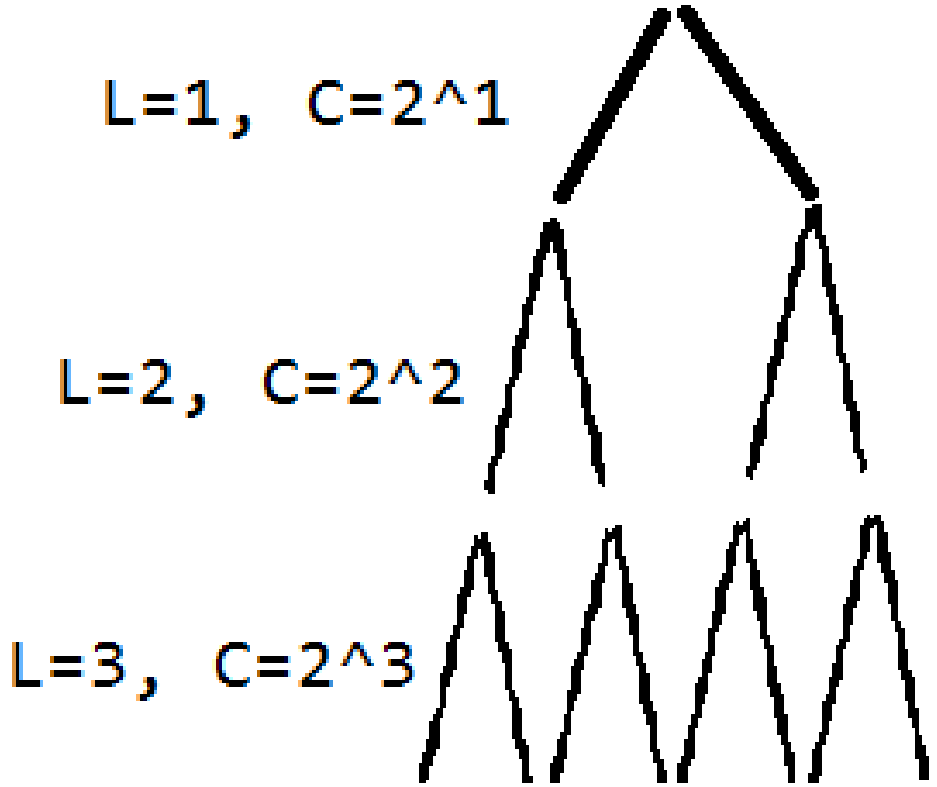
Хочу построить карточный домик, вот такой!



Нам завезли **N** карт. Сколько уровней мы сможем построить?

Нужно больше уровней, милорд!

Вот столько уйдёт на каждый уровень



А всего?

- $\boldsymbol{N} =$

$$2^1 + 2^2 + 2^3 + \cdots + 2^L =$$

$$\sum_{i=1}^{L} 2^i =$$

$$\frac{b * (1 - q^L)}{1 - q} =$$

$$\frac{2 * (1 - 2^L)}{1 - 2} = \boldsymbol{2^{L+1} - 2}$$

И чего?

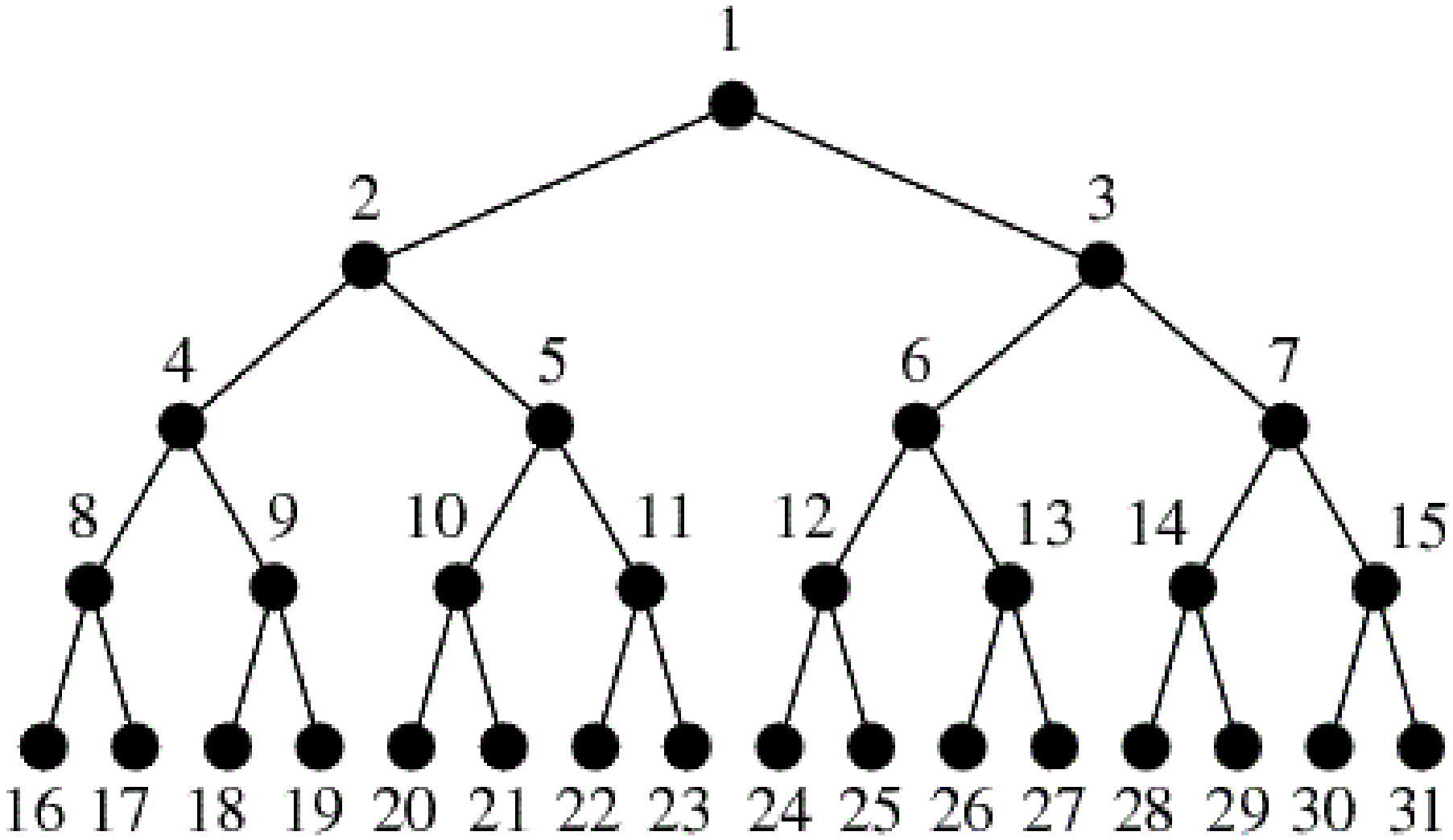
- $N = 2^{L+1} - 2$

$$N + 2 = 2^{L+1} \quad | \quad log_2$$

$$\boldsymbol{log_2(N+2) = L+1}$$

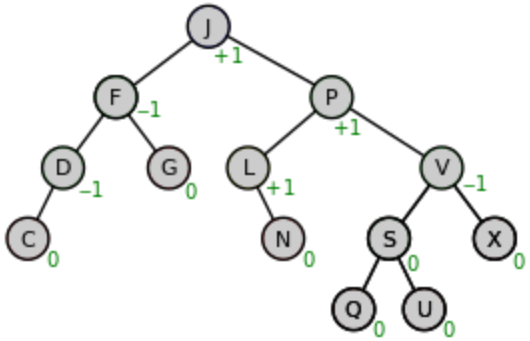
$$\boldsymbol{L = \left\lfloor log_2\left(\frac{N+2}{2}\right)\right\rfloor}$$

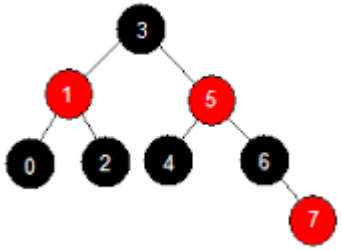
- Для 2 уровней хватит 6 карт
- Для 20 уровней нужно 2097150 карт
- Для 100 уровней потребуется уже **25353012004564588029934064107 50** карт!
- **Удвоение** числа карт добавит нам всего **один уровень**

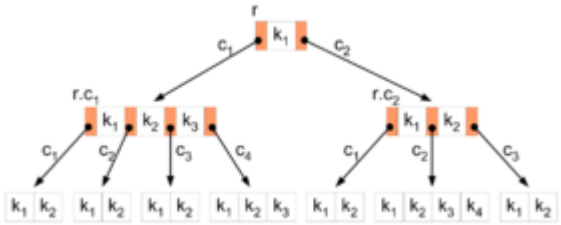

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Мы доказали, что глубина совершенного двоичного дерева пропорциональна логарифму числа его узлов

- Похожие доказательства можно провести для AVL-, RB-, B-деревьев (деревья поиска)



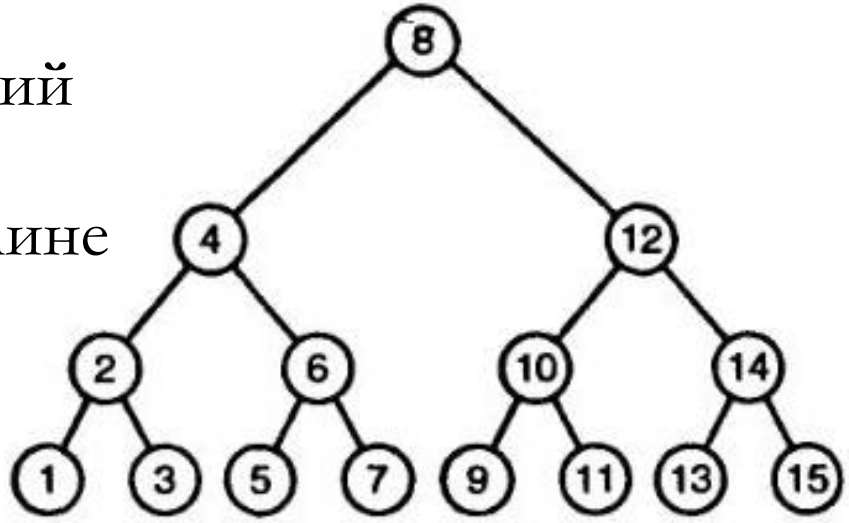




- Также, с помощью этого факта можно показать, что есть предел скорости алгоритмов сравнительных сортировок, и этот предел O(n*log(n)).

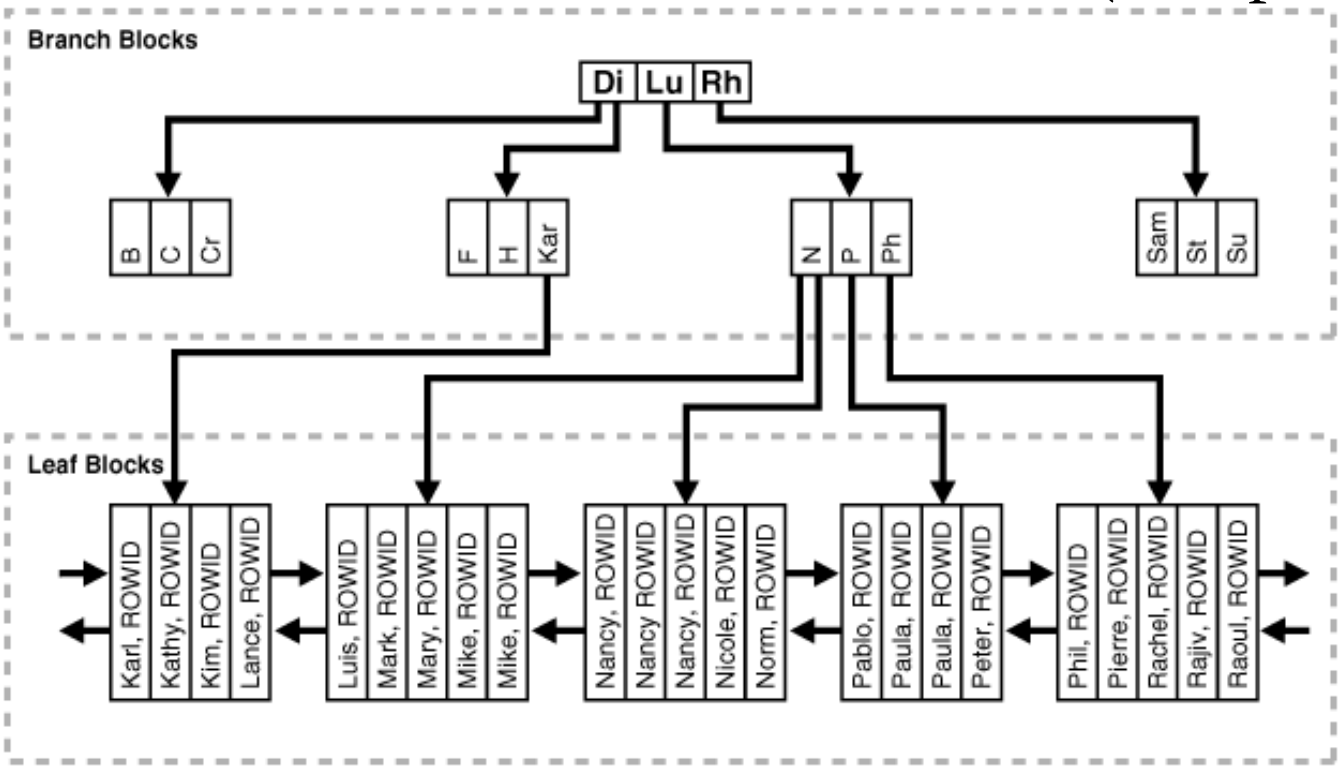
Что за деревья такие?

- Деревья поиска – деревья, у которых дети узлов упорядочены по возрастанию
- В двоичных деревьях поиска (AVL, RB) – левый ребёнок узла всегда меньше родителя, а правый – больше
- В таких деревьях время операций поиска/удаления/вставки элемента пропорционально длине пути от корня до него



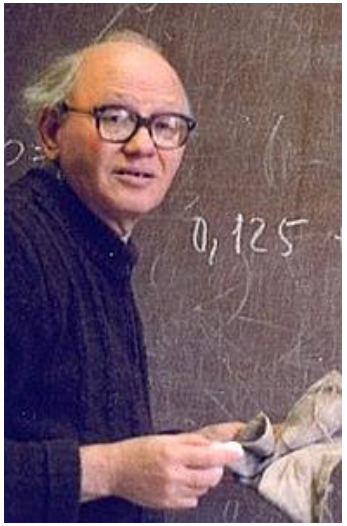
Зачем нужны такие деревья?

- Организация множеств с большим числом элементов (вставка, поиск, удаление – быстрые)
- Организация индексов в базах данных (B-деревья)



АВЛ-деревья

- Адельсон-Вельский Г.М и Ландис В.М. (1962)



- А-В – искусственный интеллект, шахматная программа Каисса (чемпион 1974)
- Ландис – работал в области дифференциальных уравнений, автор 2 учебников

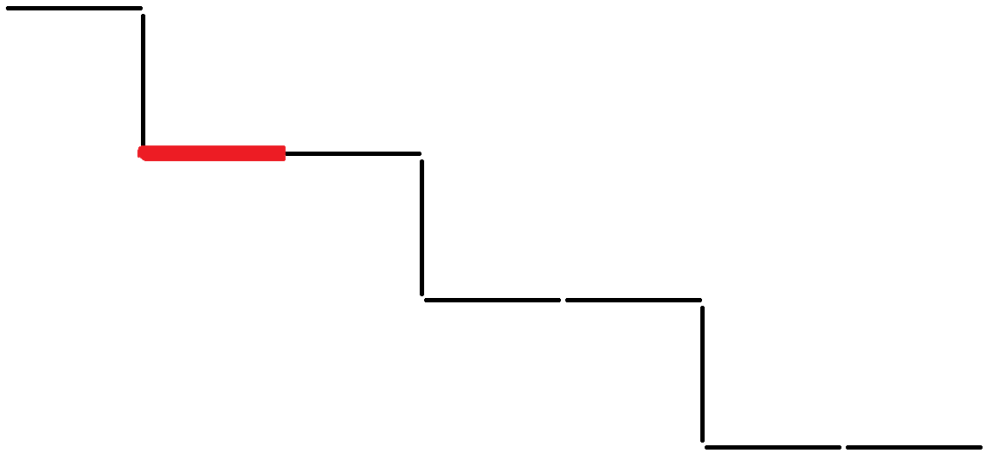
И что дальше?

- B^x Tree – в 2004 модификация B+ деревьев для эффективного хранения координат движущихся объектов
- Dancing Tree – в 2007 году Гансом Рейзером придумана модификация B+ дерева для индекса в Reiser4 (файловая система в Linux)

Угловая лестница

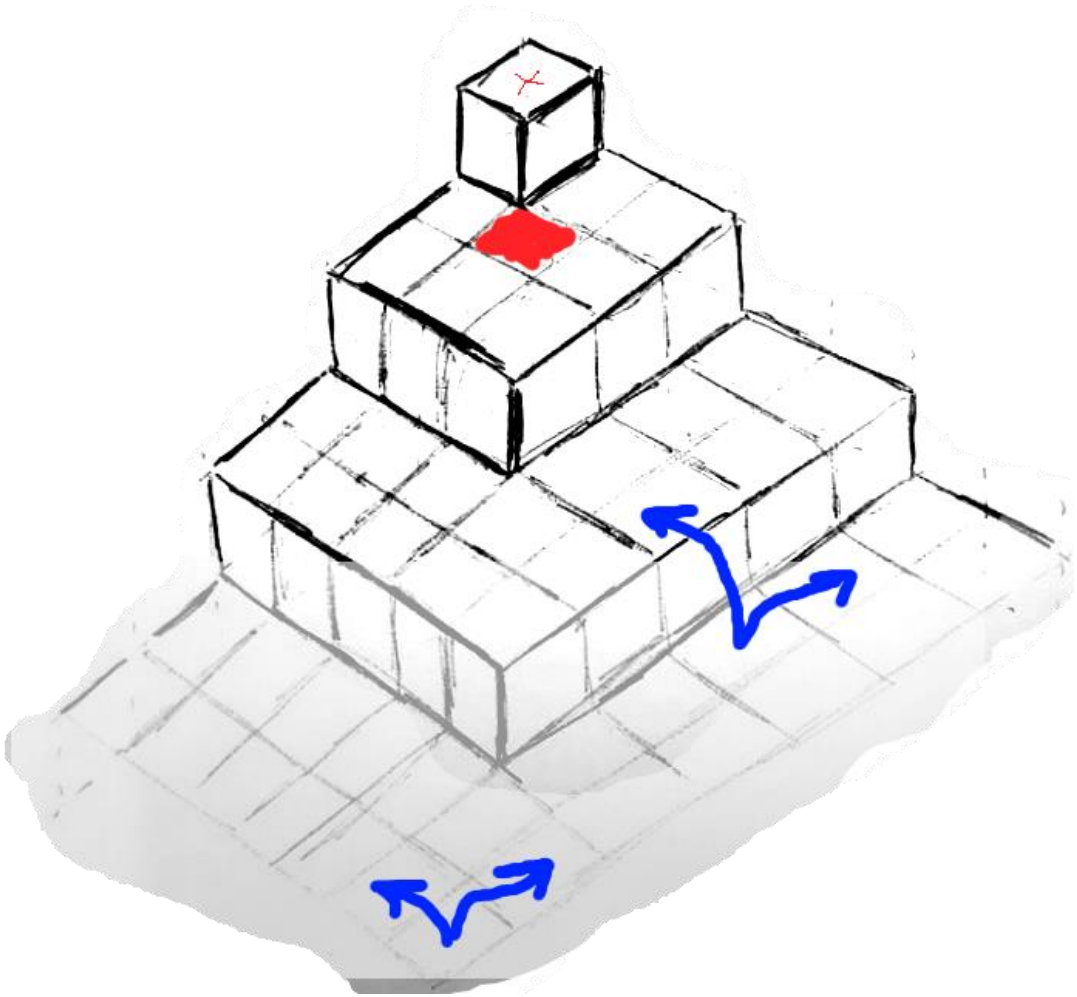
Угловая лестница

Хочу, чтобы у меня была в углу была стеклянная лестницы! Чтобы с обеих сторон выглядела одинаково! И вот такая в профиль!



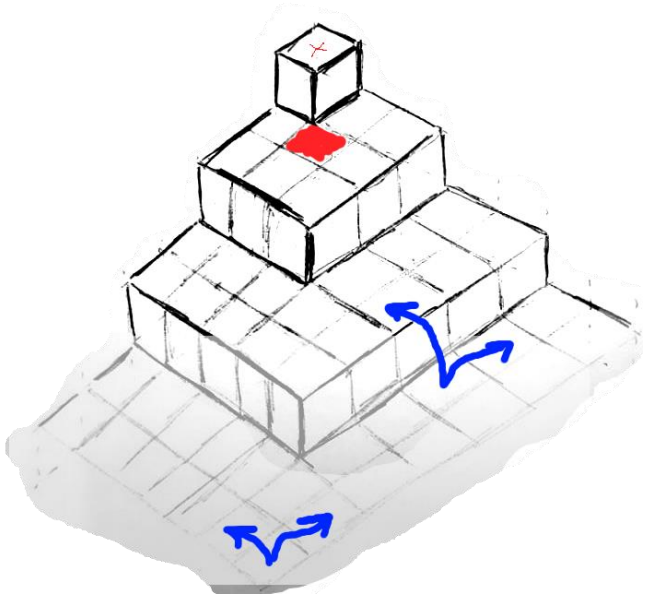
Угловая лестница 3D

Придворная забава: подняться снизу вверх, делая только шаги влево-вперёд или вправо-вперёд. Если наступаешь на красный квадратик – тебя казнят



Пара вопросов

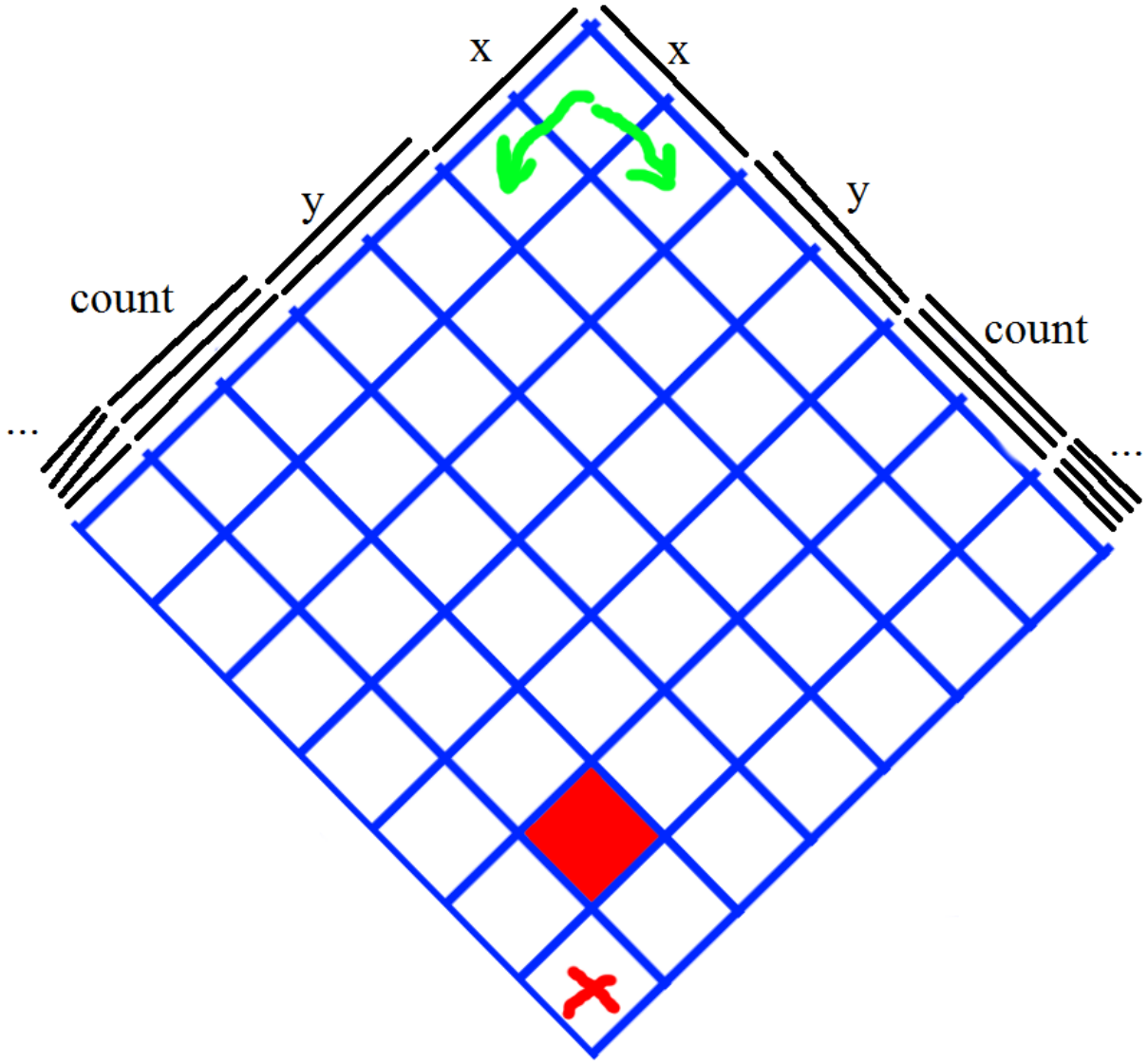
- *Существуют ли пути*, идущие через красный квадратик? Приведите пример
- *Какова вероятность* при прохождении случайным путём наступить на красный квадратик?



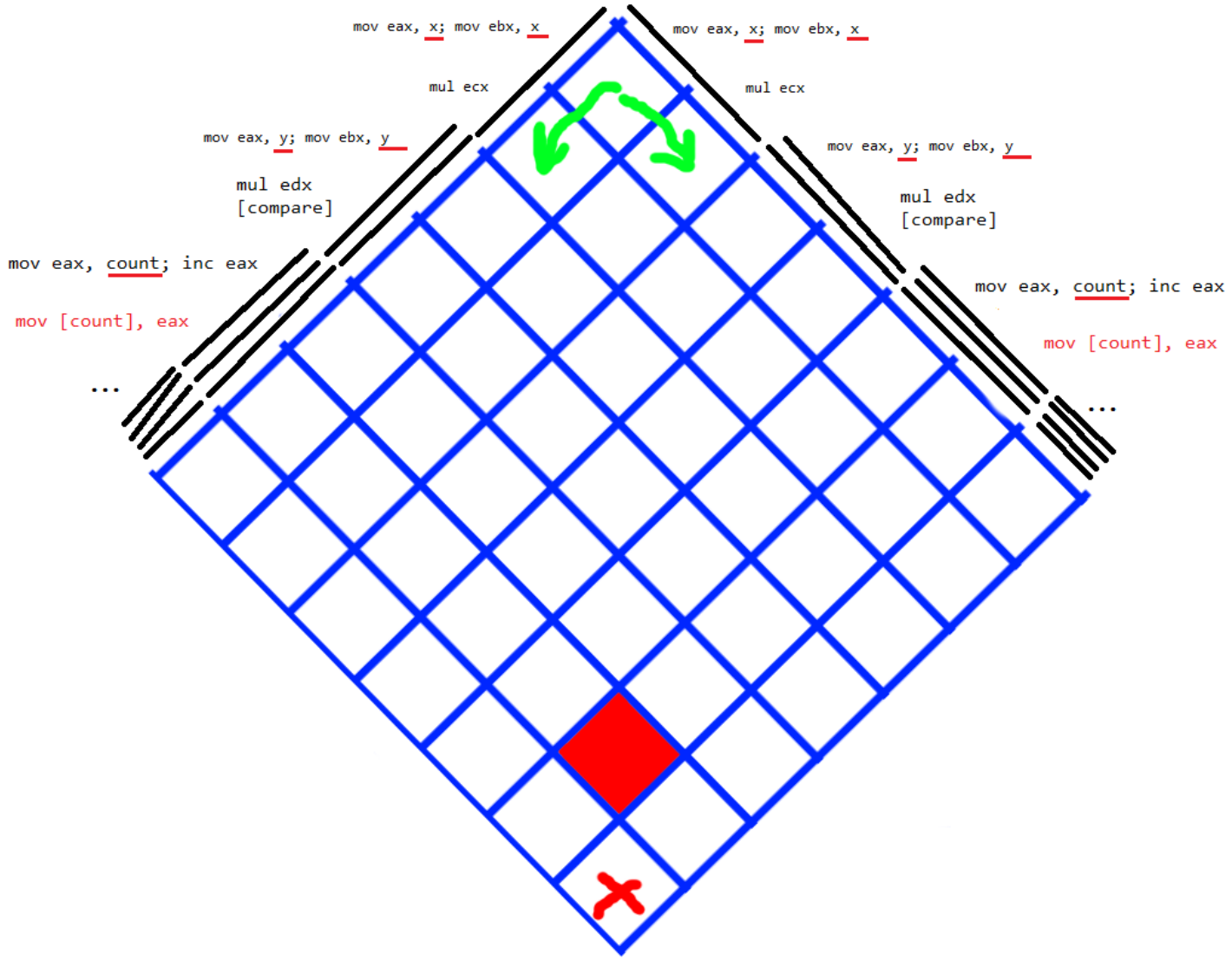
Ответы на пару вопросов

- Да, конечно, например LRLRLRLRLR
- Вероятность события – это число всех путей ведущих через красный квадратик, делённое на число всех путей по лестнице
- $P(\text{казни}) = \frac{C_{10}^{5} * C_{2}^{1}}{C_{12}^{6}} = \frac{6}{11}$

Посмотрим сверху?



Граф совместного исполнения!



Граф совместного исполнения

Модель, описывающая совместное исполнение двух функций, и позволяющая автоматически обнаруживать проблемы параллельного исполнения

- Deadlocks
- Data races
- …

Что там за мелкий шрифт?

```
mov eax, x; mov ebx, x
mul ecx
mov eax, y; mov ebx, y
mul edx
[compare]
mov  eax, count; inc eax
mov [count], eax
…
```

```
if (x*x
+
y*y

<= 1.0) {
    count++;

}
```

Что это было?

- Анализ параллельного кода – диагностика программ на потенциальные ошибки при многопоточном исполнении
- **Runtime-**анализ – диагностика программы путём запуска (Intel Parallel Studio)
- **Статический** анализ параллельного кода – диагностика **без запуска** программы

Статический анализ

Статический метод **анализа графа совместного исполнения** предложен ректором университета Иннополис Александром Тормасовым и применяется для статического анализа различных языков, поддерживающих представление LLVM

Что дальше?

- Имплементация эффективных методов статического анализа для различных проблем многопоточности
- Поддержка различных компиляторов и процессоров
- Счастье всем IDE
  - Снижение затрат компаний на исправление ошибок не очень внимательных программистов

Остались вопросы?

@sprotasov

s.protasov@innopolis.ru

https://t.me/origin_of_species