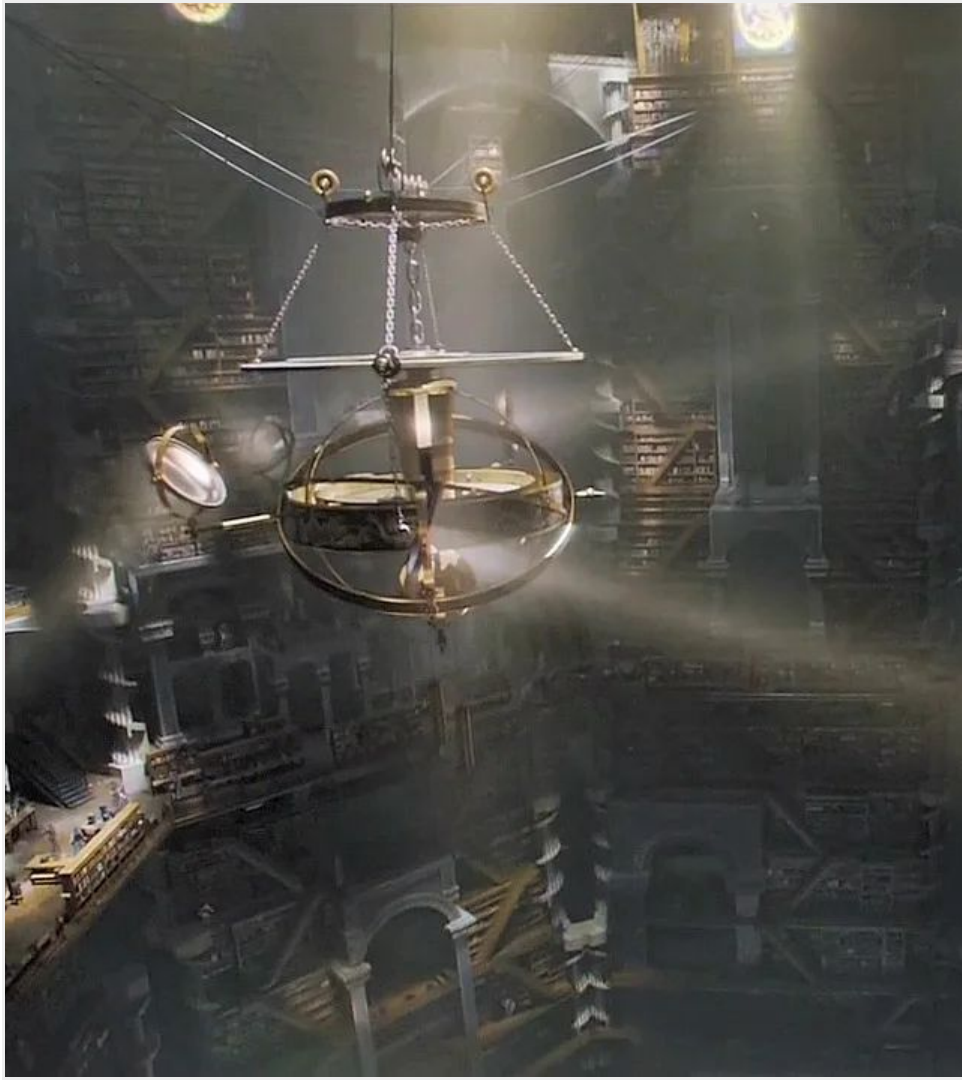
# Northwestern Russia Regional Contest

## Разбор

14 ноября 2020 года

# Задача A
# Archivist

Авторы задачи:
**Геннадий Короткевич,**
**Павел Кунявский**
Разработка задачи:
**Лидия Перовская,**
**Дмитрий Штукенберг**



# Постановка задачи

- Дан год проведения четвертьфинала
- Нужно вывести университет победитель этого года

# Решение задачи

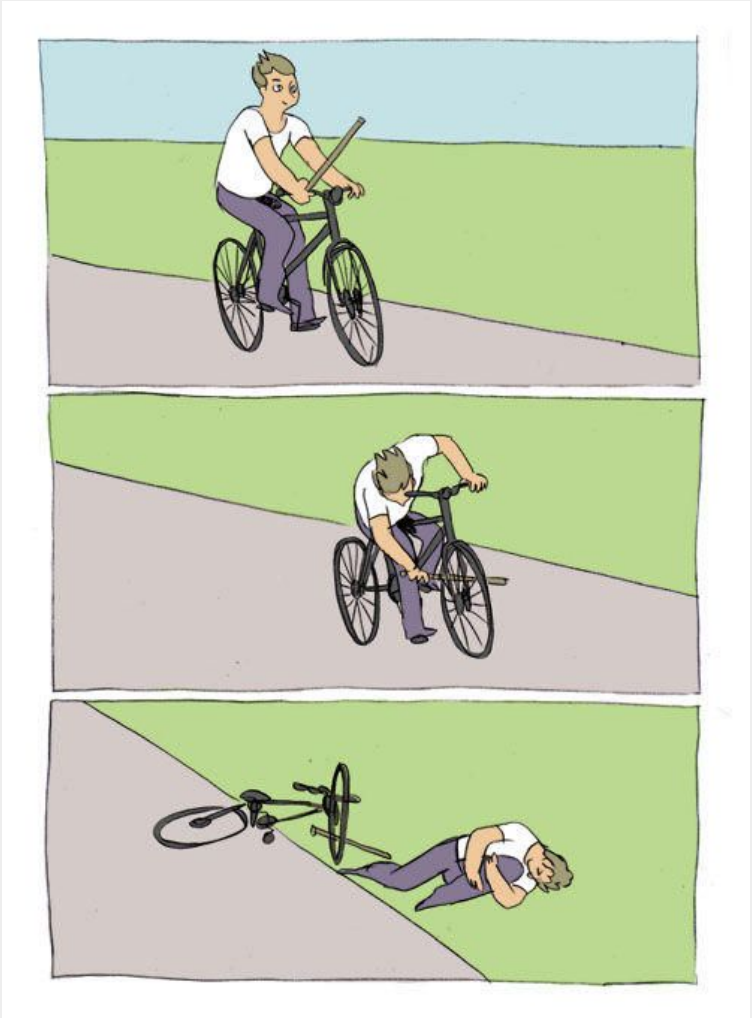
- Заведем массив с победителями в нужном порядке
- В таком случае ответ - элемента массива с индексом `[year - 1995]`
- Вместо этого сделать проверку на каждый год
- Можно заметить, что `ITMO` самый частый ответ и уменьшить количество проверок
- Лучше копировать тексты, чтобы не опечататься

# Код решения на Python

```
y = int(input())
if y == 2006:
    print('PetrSU, ITMO')
elif y in [1996, 1997, 2000, 2007, 2008, 2013, 2018]:
    print('SPbSU')
else:
    print('ITMO')
```

# Задача B
# Bicycle

Автор задачи:
**Геннадий Короткевич**
Разработка задачи:
**Виталий Аксенов**



# Постановка задачи

- Есть два тарифа аренды велосипеда.
- У каждого есть цена. бесплатное время в день и стоимость минуты сверх этого времени
- Надо найти сколько придется заплатить за оба тарифа, если ездить T минут каждый рабочий день

# Разбор случаев

- Можно разобрать три случая.
  - Если количество минут не больше 30, то ответ a и b
  - Если количество минут от 30 до 45, то ответ:
    a + (T - 30)*21*x и b
  - Если количество минут больше 45, то ответ:
    a + (T - 30)*21*x и b + (T - 45) * 21 * y

# Использование функции max

- Можно записать без разбора случаев, используя функцию max
  - Для первого тарифа ответ a + max(0, T - 30) * 21 * x
  - Для второго тарифа ответ b + max(0, T - 45) * 21 * y

# Задача C
# Corrupted Sort

Автор задачи:
**Михаил Дворкин**
Разработка задачи:
**Борис Минаев**

# Постановка задачи

- Нужно отсортировать $n$ чисел с помощью операций «Сравнить числа на позициях $i < j$, и поменять их местами, если $a[i] > a[j]$»
- Раз в $2n$ операций случайная пара чисел меняется местами

# Решение задачи

- Если бы дополнительных операций не было, можно было бы использовать почти любой известный алгоритм сортировки
- Чтобы побороться со случайными изменениями, придумаем алгоритм, который за последовательные $2n$ действий «отменяет» случайное действие, а также совершает дополнительную полезную работу

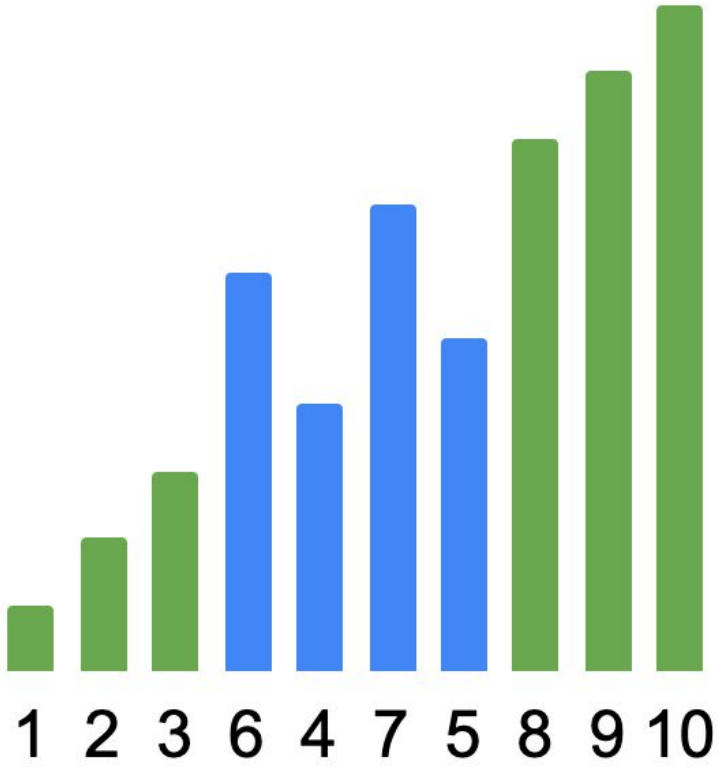
# Решение задачи

```
while True:
  for i in list(range(1, n)):
    print(i, i + 1)
  for i in list(range(n - 1, 0, -1)):
    print(i, i + 1)
```

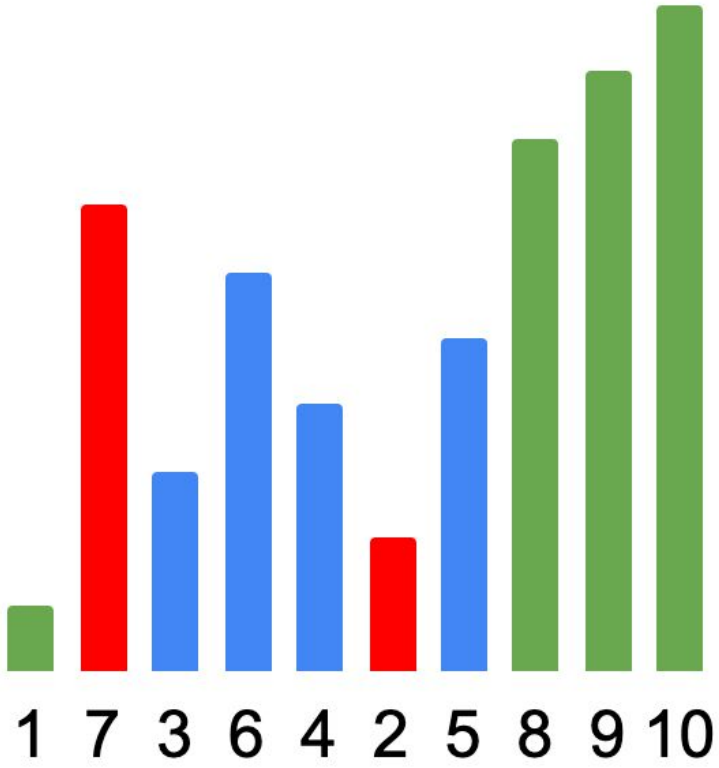
# Почему это работает?

- Пусть прямо перед случайной операцией $L$ минимальных чисел и $R$ максимальных чисел уже стоят на своих позициях
- Через $2n$ операций $L$ и $R$ точно не уменьшатся, а с довольно большой вероятностью увеличатся
- Когда $L + R$ станет равно $n$, массив будет отсортирован

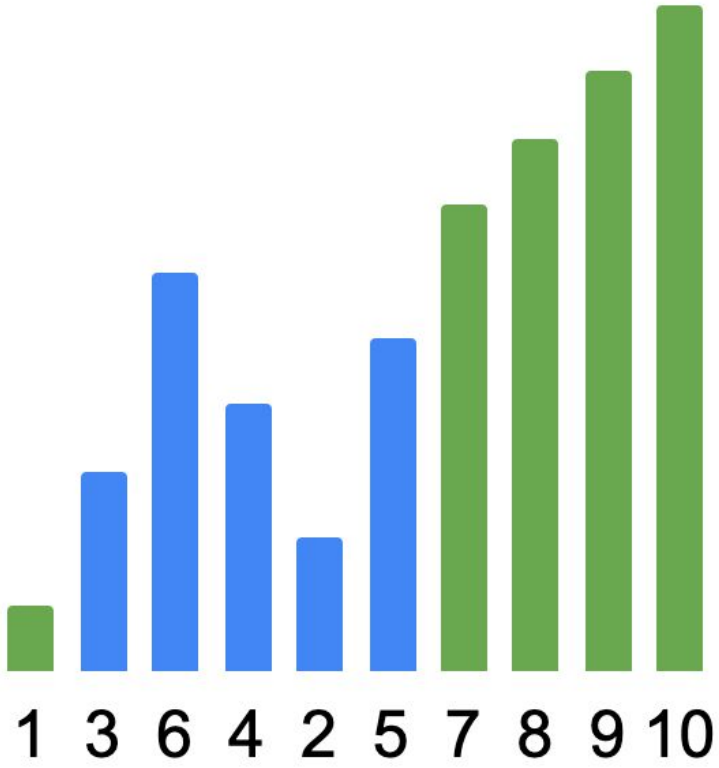
# До случайной операции (L = 3; R = 3)



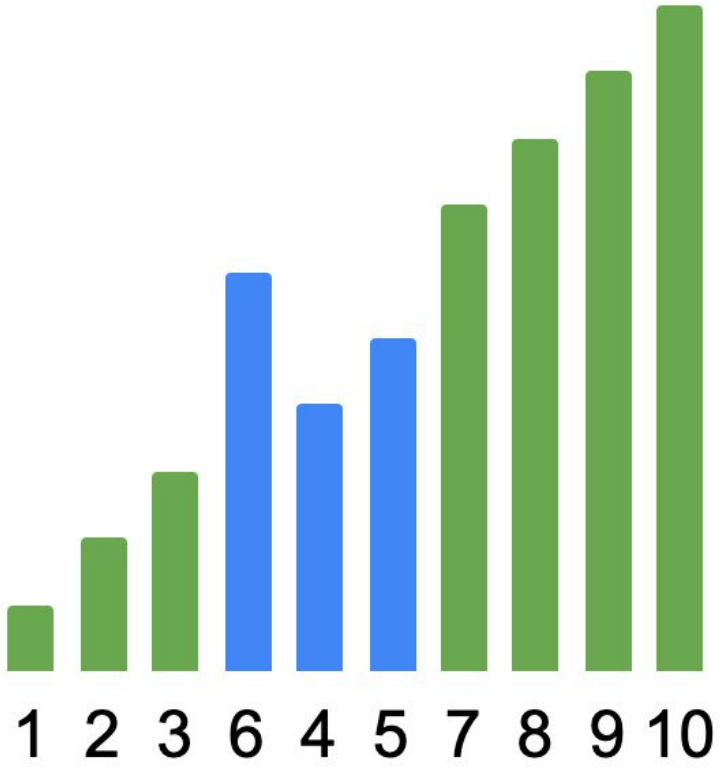
# После случайной операции



# После прохода слева направо



# После прохода справа налево (L = 3; R = 4)



# Почему это работает?

- Если случайная операция не затронула первые $L$ чисел, то через $2n$ операций $L$ увеличится как минимум на один
- Если случайная операция не затронула последние $R$ чисел, то через $2n$ операций $R$ увеличится как минимум на один

# Задача D
# Display

Авторы задачи:
**Георгий Корнеев**
Разработка задачи:
**Георгий Корнеев**

# Постановка задачи

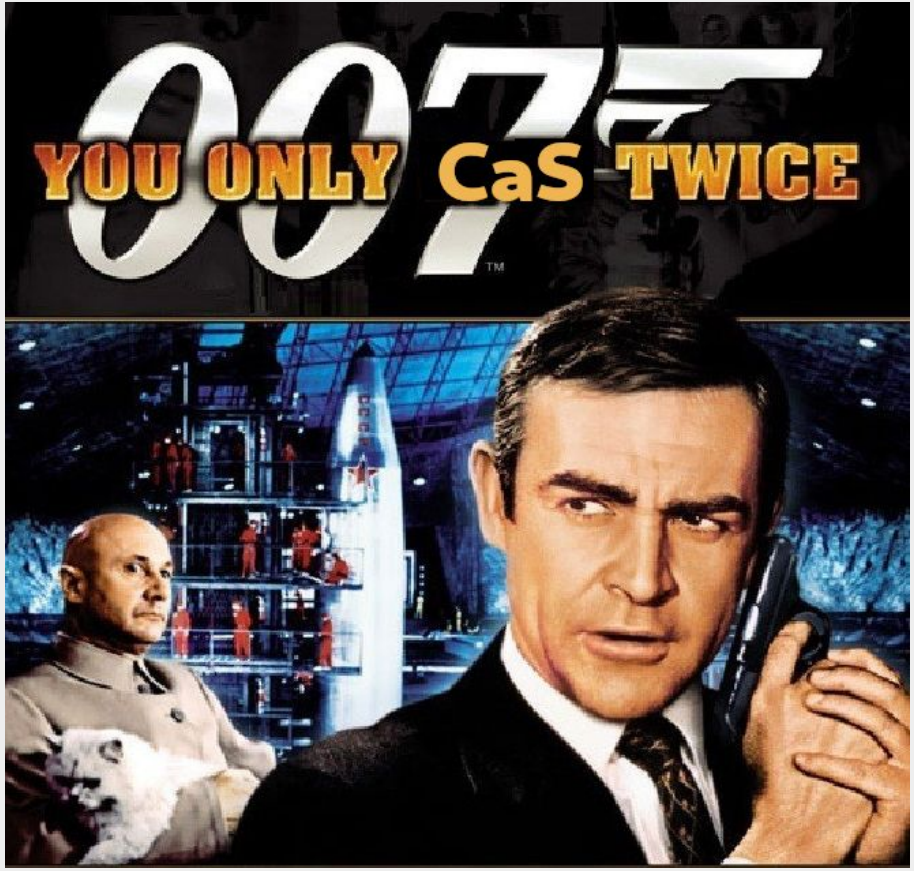
- Дан шрифт, в котором все буквы имеют размер *w* x *h* пикселей
- Есть дисплей для бегущей строки высоты *h*
- Пиксель на дисплее ломается после *s* включений/выключений
- Найти кратчайший текст, при запуске которого на дисплее сломается хотя бы один пиксель

# Решение задачи

- Посмотрим на конкретную строку на дисплее
- Со всеми пикселями на ней произойдёт одинаковое число включений/выключений
- Сколько?
  - Для каждой буквы в тексте -- число отрезков подряд идущих включенных пикселей (‘#’)
  - Сумма
- Выберем символ, в котором таких отрезков больше всего, и повторим его в тексте нужное число раз
- Возьмём максимум по всем строкам дисплея

# Задача E
# Easy
# Compare-and-Set

Автор задачи:
**Виталий Аксёнов**
Разработка задачи:
**Сергей Мельников**



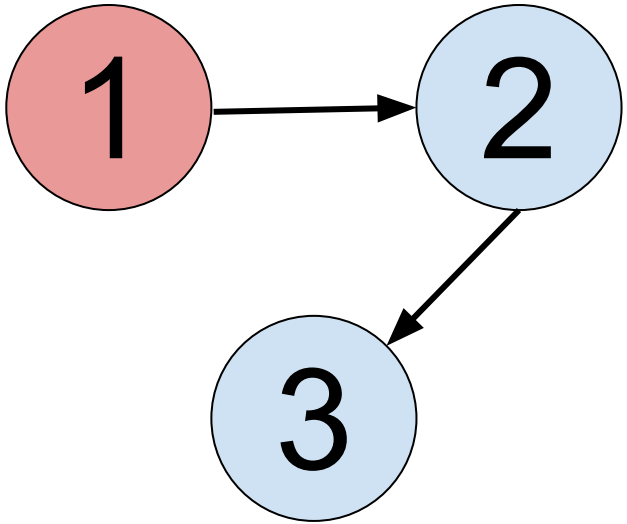
# Постановка задачи

- Дана глобальная переменная ***c*** и множество операций Compare-and-Set и результатом выполнения каждой
- ```
  CompareAndSet(a, b):
      if a == c:
          c = b
          return 1
      else:
          return 0
  ```
- Найти порядок, в котором выполнение операций приводит к нужному результату каждой
  ```

# Решение задачи

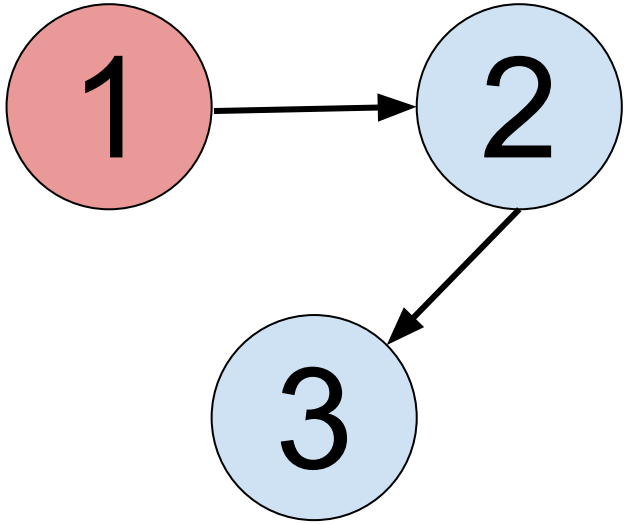
- Построим граф, где вершины — значения переменной ***c***, а рёбра — успешные операции `CompareAndSet`.

```
4 1
1 2 0
1 2 1
2 3 1
3 4 0
```



# Решение задачи

- Найдём в этом графе путь начинающийся в вершине ***c***, и проходящий по каждому ребру ровно один раз
  - Эйлеров путь
- Если пути не существует, то решения нет



# Решение задачи

- Неуспешные операции
  - Если a ≠ c
    - сделаем перед первой успешной
  - Если a = c
    - сделаем, как только окажемся в другой вершине
    - если все успешные операции имеют вид CompareAndSet(c, c), то решения нет

# Задача F
# Futures Market Trends

Автор задачи:
**Федор Царев**
Разработка задачи:
**Павел Кунявский**

# Постановка задачи

- Трендом называется отрезок, у которого среднее изменение больше стандартного отклонения изменений хотя бы в $C$ раз
- Дан массив, надо найти все подотрезки, являющиеся трендами

# Медленное решение

- Сразу перейдем к массиву разностей соседних
- Решение за $O(n^3)$:
- Переберем все подотрезки и посчитаем по формуле из условия
- Нужно научиться пересчитывать среднее и отклонение при переходе от $[i, j]$ к $[i, j + 1]$

# Решение задачи

- Будем поддерживать сумму чисел S и сумму их квадратов T
- Числитель и знаменатель можно вычислить в целых и проблем с точностью не будет

$$\frac{A}{D} = \frac{\frac{1}{n}S}{\sqrt{\frac{1}{n}\sum(s_i - \frac{1}{n}S)^2}} = \frac{S}{\sqrt{\sum(ns_i^2 - 2nSs_i + \frac{1}{n}S^2)}} = \frac{S}{\sqrt{n \cdot T - S^2}}$$

# Задача G
# Grammar Path

Автор задачи:
**Михаил Дворкин**
Разработка задачи:
**Илья Збань,**
**Михаил Дворкин**

# Постановка задачи

- Дана контекстно-свободная грамматика и ориентированный граф с терминалами на рёбрах
- Найти кратчайший путь из $s$ в $t$, текст вдоль которого, — это слово из грамматики

# Решение: для строки

- Нужно обобщить алгоритм Кока—Янгера—Касами
- $a[i][j][k]$ — верно ли, что отрезок слова $s[i..j]$ порождается нетерминалом $k$.
- Если отрезок длины один, то можно, если есть такая продукция
- Иначе переберем как разбить на две части, и по какой продукции сделать переход
- Пересчитываем от коротких к длинным
- Общее время работы $O(n^3 p)$

# Решение: для произвольного графа

- Считаем $d[v_1][v_2][k]$ — кратчайший путь из $v_1$ в $v_2$ который соотвествует нетерминалу k.
- Если есть ребро и такая продукция, то 1
- Иначе переберем продукцию и промежуточную вершину
- В каком порядке перебирать состояния?
    - ≈ Алгоритм Дейкстры (следующий слайд)
- Неясно, насколько большой может быть ответ
    - Известные верхняя и нижняя оценка расходятся очень сильно.
    - Есть пример на $2^{|N|\times n}$ и оценка на $2^{|N|\times n^2}$

# Решение: обобщаем алгоритм Дейкстры

- Пусть мы знаем, что ответ для $dp[v_1][v_2][k_1]$ и $dp[v_2][v_3][k_2]$ точно правильный
- Можно обновить $dp[v_1][v_3][p]$, если есть правило $p \to k_1 k_2$
- Самое маленькое необработанное значение правильное
- Попробуем приклеить к нему все что можно с обеих сторон
- Когда будем обрабатывать большую по длине из двух частей, посчитаем правильный ответ
- Итого: решение работает за время $O(n^3 p \times \log(n) \times ???)$

# Задача H
# Heroes of Coin Flipping

Автор задачи:
**Артем Васильев**
Разработка задачи:
**Артем Васильев**

# Постановка задачи

- Большой турнир на выбывание, любой участник побеждает каждого с вероятностью ½
- Смотрим сначала некоторые матчи в турнирной сетке
- Потом все остальные в случайном порядке
- В скольки матчах мы не будем знать победителя?

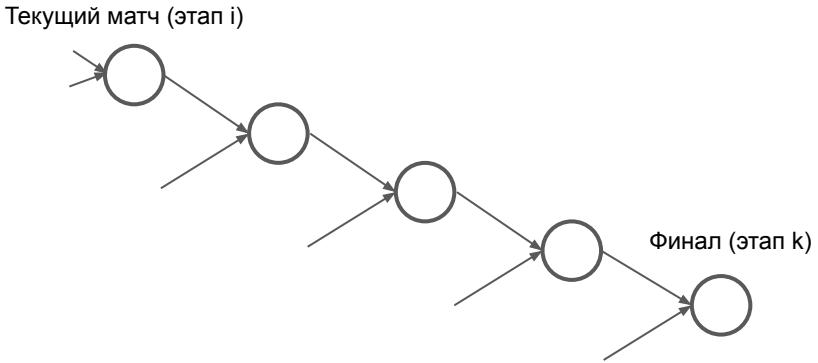
# Теория вероятности

- Турнирная сетка - полное двоичное дерево
- Математическое ожидание линейно
- Достаточно для каждого матча найти вероятность того, что он будет зрелищным
- Мы *не должны видеть победителя* текущего матча в *более позднем* матче

# Случай n = 0

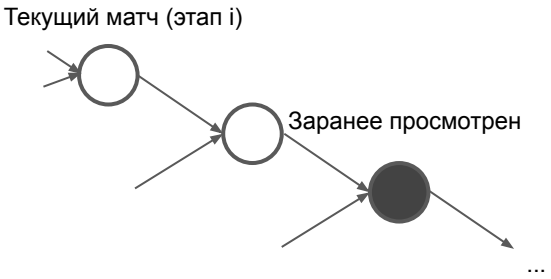
- С какой вероятностью мы не видели победителя текущего матча?
- Пусть он проиграл сразу в следующей игре
  - ½ (проиграл в следующем матче) * ½ (текущий матч мы посмотрели раньше)
- Пусть выиграл один, но проиграл после него
  - ½ * ½ (выиграл и проиграл) * ⅓ (текущий матч мы посмотрели раньше двух других)
- …
- Учесть, что он может выиграть всё

# Случай n = 0




- 0 выигрышей после текущего матча: ½ * ½
- 1 выигрыш: ½ * ½ * ⅓
- 2 выигрыша: ½ * ½ * ½ * ¼
- …
- Проигрывает в финале: $½^{k-i} * 1 / (k - i + 1)$
- Побеждает всех: $½^{k-i} * 1 / (k - i + 1)$

# Случай n > 0




- Сумма такая же, как в прошлом случае
- Но если на пути встретился просмотренный матч, то сумма обрывается
- Зависит только от расстояния до просмотренного матча

# Предобработка

- Для матча на глубине 0, 1, …, k-1 предпосчитаем вероятность зрелищности
- Два случая:
  - Мы дошли до предпросмотренного матча
  - Мы дошли до корня
- Все n заданных во входе матчей обрабатываем аналогично (не учитывая вероятность посмотреть в неправильном порядке)

# Решение

- Запустим DFS по дереву турнира
- Если текущее поддерево пустое (в нем нет предпросмотренных матчей), то ответ для него можно посчитать за O(k)
  - Пустых поддеревьев O(n)
- Для промежуточных вершин также берем предподсчитанное значение
  - Таких вершин всего O(nk)
- Суммарное время работы O(nk)

# Задача I
# Integer Square

Авторы задачи:
**Павел Маврин**
Разработка задачи:
**Павел Маврин**

# Постановка задачи

- Построить квадрат площади $s$
- С целочисленными координатами вершин

# Решение

- Площадь квадрата равна $c^2$, где $c$ - длина его стороны
- Поставим одну вершину квадрата в (0, 0)
- Пусть другая вершина находится в ($x$, $y$)
- Тогда $c^2 = x^2 + y^2$, по теореме Пифагора

# Решение

- Переберем *x*, *y* от 0 до 100, поищем пару, где $x^2 + y^2 = s$
- Если не нашли, то ответ “*Impossible*”
- Иначе выведем квадрат с помощью вектора (x, y), повернутого 4 раза
  - (0, 0)
  - (x, y)
  - (x - y, y + x)
  - (-y, x)

# Задача J
# Joint Password Storage

Авторы задачи:
**Юрий Петров,**
**Михаил Дворкин**
Разработка задачи:
**Павел Кунявский**

# Постановка задачи

- Дан пароль от 10 до 50 символов из букв и цифр
- Надо представить его как побитовый ксор нескольких строк
- Каждая строка должна быть верным арифметическим равенством

# Решение: когда NO?

- Бит 64 есть во всех буквах. Но в выражениях есть только в [] и {}.
- Скобок в выражении всегда четно, поэтому букв должно быть четно
- Из битов 64 и 32 есть ровно один везде кроме {}.
- Если строка четной длины, то бита 32 всегда будет четно во всей строке.
- Значит в четных строках четно отдельно больших и маленьких букв
- Все остальное можно!

# Решение: случайные строки + Гаусс

- Хотим найти набор из 7*n-1 или 7*n-2 линейно независимых выражений
- Если сможем, то найти ответ для одного пароля можно методом Гаусса
- Как искать базис? Берем случайные элементы множества пока не хватит.

# Решение 1: генерация случайных равенств

- Хотим генерировать случайные выражения
- Нужно, чтобы были разные длины
- Нужно, чтобы были не слишком большие значения
- Когда совпали значения, собираем равенство

# Решение 1: генерация случайных равенств

- Генерировать можно по грамматике, делая переходы с разными вероятностями
- Числа надо брать маленькие, но не всегда из одной цифры
- Есть очень маловероятные биты (например скобка на 2 и предпоследней позиции). Их можно добавить руками

# Решение: явная конструкция, выравниваем 32

- Можно без случайности
- Для начала сделаем, чтобы бит 32 был везде одинаковый.
  - [[0]]*1...1=0 - на соседних с краями
  - [0]*1...1=0 - на краях
  - 1...1*[0]=1..1*0 - позволяет двигать
  - Так можно починить кроме 3-4 с концов
  - 0=[[1...1]]*0 и 0=[11...11]*0 позволяют перенести проблему налево
  - 11*[0]=0*1…1 и 0*[11]=0*1...1 позволяют починить их

# Решение: явная конструкция

- Точно также как выравнивали бит 32 зануляем бит 64
- Зануляем бит 32, добавляя что угодно без [] и {}
- Будем добавлять по 2 строчки вида 0=0*... или симметричных. Подбором цифр и знаков умножения можно обнулить все, кроме бита 16 на крайних позициях
- Его можно обнулить строками (0)=0*...

# Решение: явная конструкция базиса + Гаусс

- Можно выбрать промежуточный вариант между решениями
- Берем все строки из второго решения
- Находим среди них базис
- Методом Гаусса находим разложение на строки из нашего множества

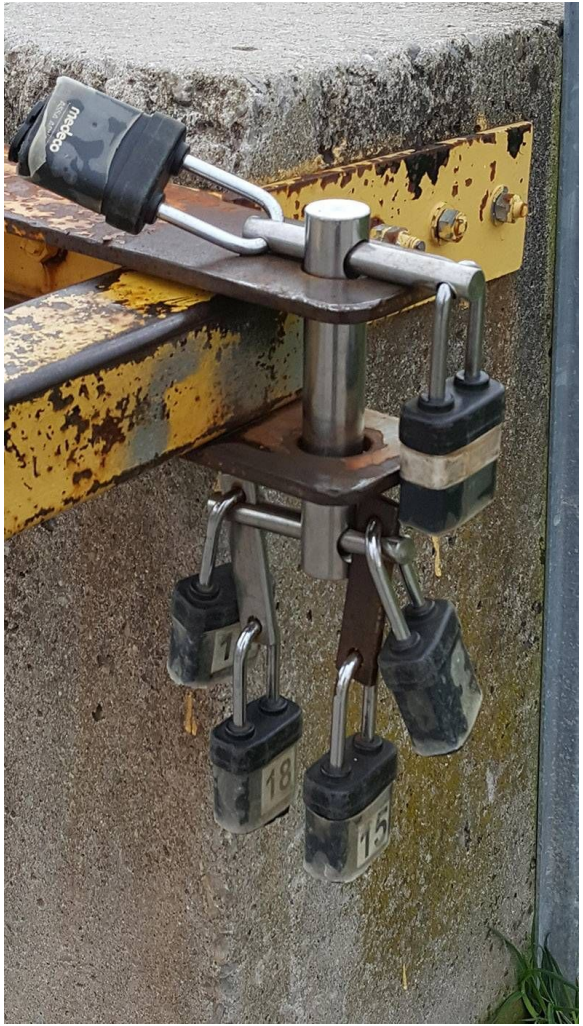
# Задача K
# Keys and Locks
# Boolean Logic

Автор задачи:
**Михаил Дворкин**
Разработка задачи:
**Михаил Дворкин**

# Постановка задачи

- Дана булева формула «какие подмножества друзей можно пускать в гараж»
- Реализовать её с помощью замков и ключей
- … в псевдографике!
- … уложившись в размер 50×50!

логическое «ИЛИ» в реальности →

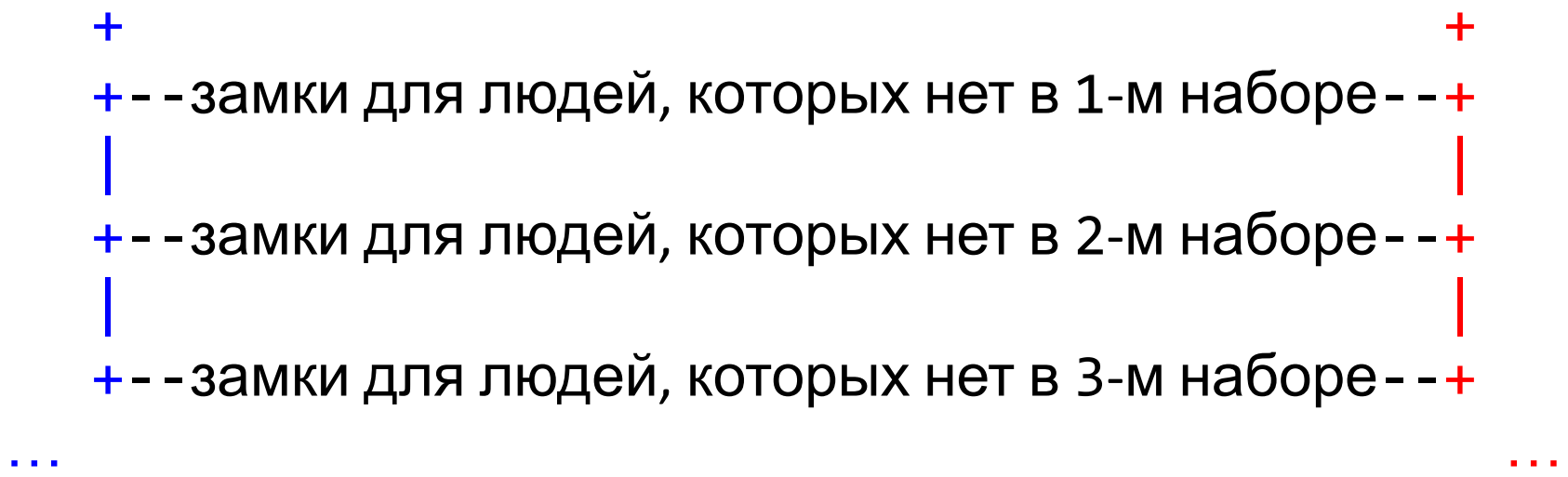


# Решение без ограничения на размер поля

- Булева функция должна быть монотонной:
  - f(0, 0, 1, 1, 0) = 1, могут войти
  - f(0, 1, 1, 1, 1) = 0, не могут войти…
    - ну так не суйте ключи!
- Любая монотонная функция — выразима
- Она выражается через and и or, которые рисуются так:
  - and = параллельное соединение
  - or = последовательное соединение

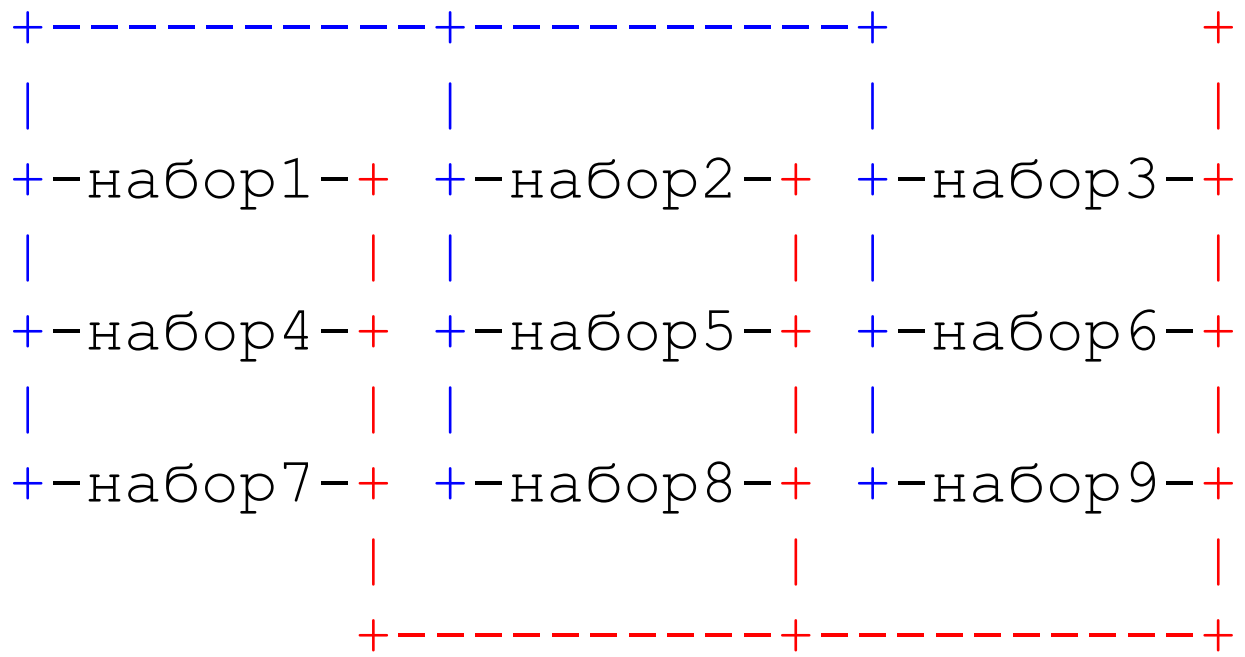
# Решение, влезающее в 512×10

- (Это совершенная конъюнктивная нормальная форма)
- Рассмотрим все наборы, которым нельзя войти



# Решение, влезающее в 50×50

- Если есть два запрещённых набора $A \subset B$, то нужно только соединение, запрещающее $B$
  - Не более $C_8^4 = 70$ наборов



# Задача L
# Lost Permutation

Автор задачи:
**Артем Васильев**
Разработка задачи:
**Артем Васильев**

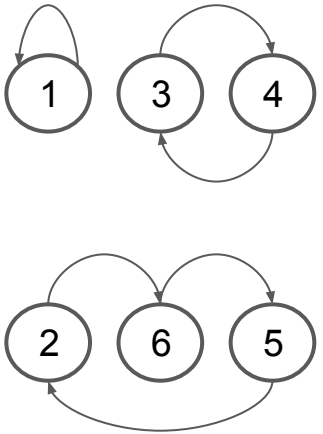
# Постановка задачи

- Интерактивная задача
- Загадана перестановка $\pi$
- Можно сделать два запроса
- По перестановке f узнать перестановку $\pi^{-1} \circ f \circ \pi$

# Свойства перестановок

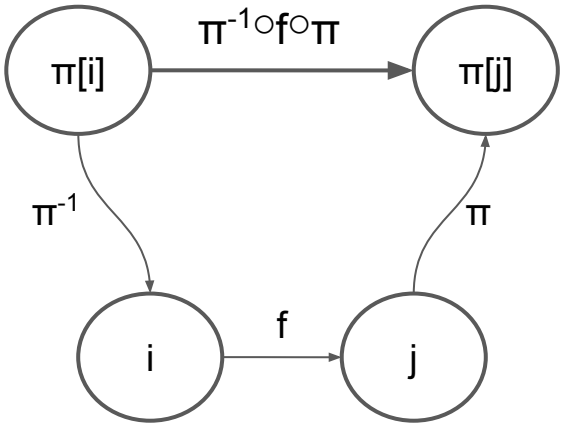
- Полезно понимать свойства композиции перестановок
- Перестановка как орграф i -> a[i] = набор циклов
- Композиция $a \circ b$ соответствует операции “пройти по ребру из a, затем по ребру из b”

(1, 6, 4, 3, 2, 5)



# Сопряжение относительно π

- Операция, описанная в условии, называется *сопряжением* f относительно π
- Пусть у нас было ребро i -> j в перестановке f
- Во что оно превратится в $\pi^{-1} \circ f \circ \pi$ ?
- π[i] -> π[j]



# Циклы

- Получается, что элементы на циклах в b “переименовываются” по перестановке π
- Цикл длины k перейдет в цикл длины k
- Вместо цикла a -> b -> c -> … будет цикл π[a] -> π[b] -> π[c] -> ...

# Решение

- Спросим цикл длины n:
  - [2, 3, 4, …, n, 1]
- В ответ получим тоже цикл длины n, но мы не знаем, откуда его нужно начать читать
- Вторым запросом выделим 1 в отдельный цикл:
  - [1, 3, 4, …, n, 2]
- Найдя неподвижную точку (a[i] = i) в ответе, знаем π[1]
- Если прочесть первый цикл, начиная с π[1], получится π[2], … π[n]

# Задача M
# Mind the Gap

Автор задачи:
**Нияз Нигматуллин**
Разработка задачи:
**Нияз Нигматуллин**

# Постановка задачи

- Игра: у каждого есть карточка с числом
  - заданы **n** карточек
- Каждый действует по стратегии:
  - выкладывает, когда его число не более чем на **d** больше числа на столе
- Какое **d** выбрать, чтобы все карты выложились в возрастающем порядке?

# Наблюдения

- Упорядочим все карты по возрастанию
  - получим массив $a_1, a_2, \ldots, a_n$
- Изначально на столе число 0, поэтому $a_0 = 0$
- Чтобы каждый мог выложить карту
  - $a_i - a_{i-1} \leqslant d$
- Если $a_i - a_{i-2} \leqslant d$, после того, как выложат $a_{i-2}$:
  - двое захотят выложить $a_{i-1}$ и $a_i$

# Решение

- Критерий: для всех **i**: $a_i - a_{i-1} \leqslant d$, и для всех **i**: $d < a_i - a_{i-2}$
- Ответ: любое число $[\max\{a_i - a_{i-1}\} .. \min\{a_i - a_{i-2}\} - 1]$
  - Если множество пустое, **d** не существует
- Можно выбрать $d = \max\{a_i - a_{i-1}\}$, проверить $a_i - a_{i-2} > d$

# Задача N
# Nunchucks Shop

Автор задачи:
**Павел Маврин**
Разработка задачи:
**Павел Маврин**

# Постановка задачи

- Построить минимальный набор строк длины $n$ из 0 и 1
- Нужно уметь представлять любую строку длины $2n$, в которой ровно $k$ единиц, как конкатенацию двух строк из набора

# Решение

- Посмотрим, какие нам нужны половинки
- Нужны все половинки с *x* единицами, если $x <= k$, $x <= n$ и $k - x <= n$
  - Дополнительно, если *k* четное, то каждую строку из *k/2* единиц нужно взять дважды
- Перебираем все x = 0..k
- Если x <= n и k - x <= n, добавляем к ответу число половинок длины *n* с *x* единицами с учётом симметрии

# Как учесть симметрию?

- Учтем, что половинки можно переворачивать
- Все половинки, кроме палиндромов, разбиваются на пары, из каждой пары достаточно взять только одну
- Если $n$ четно и $k$ нечетно, то палиндромов нет
  - В таком случае ответ $C(n, k) / 2$
- Иначе есть $C(\lfloor n / 2 \rfloor, \lfloor k / 2 \rfloor)$ половинок, являющихся палиндромами
  - Тогда ответ $(C(n, k) + C(\lfloor n / 2 \rfloor, \lfloor k / 2 \rfloor)) / 2$

```
printf("%s\n", "Спасибо за внимание")
```

Материалы олимпиады

https://nerc.itmo.ru/