## Первая задача

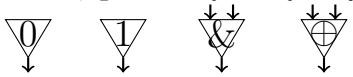
*Программой* вычисления булевой функции над переменными $x_1, \ldots, x_n$
назовём последовательность команд вида $\boxed{x = f(y_1, \ldots, y_k)}$:
вычислить в $x$ суперпозицию функций, вычисленных ранее в $y_1, \ldots, y_k$
(в $x_1, \ldots, x_n$ заранее вычислены соответствующие тождественные функции).
*Стоимость* команд $\boxed{x = \overline{y}}$ и $\boxed{x = y \oplus z}$ — 0. Стоимость команд $\boxed{x = y \,\&\, z}$ — 1.
Стоимость программы — это суммарная стоимость её команд.
Стоимость булевой функции $f$ — это стоимость самой дешёвой программы,
в которой хотя бы раз вычисляется функция $f$.
*Пример*: функция, реализуемая полиномом $x_1 \oplus x_2 \oplus x_3$, бесплатна:

$$y = x_1 \oplus x_2; \quad z = y \oplus x_3$$

**?** Сколько сто́ит медиана $(m)$? $m(x_1, x_2, x_3) = x_1x_2 \oplus x_1x_3 \oplus x_2x_3$
Сколько сто́ит самая дорогая функция от $n$ переменных?

## Вторая задача

Рассмотрим булеву функцию $f(x_1, \ldots, x_n)$
и схему из функциональных элементов, реализующую $f$, в таком базисе:




Предположим, что в схеме может *сломаться* **не более чем один** элемент,
и что сломанный элемент всегда выдаёт неправильное (противоположное) значение.
Для проверки исправности схемы будем проводить *эксперименты* такого вида:
подаём значения на входы схемы, получаем значение на выходе,
сравниваем его с соответствующим значением функции $f$.
*Стоимость* схемы — это наибольшее число экспериментов,
требуемое для проверки её исправности (в худшем случае).
Стоимость функции $f$ — это стоимость самой дешёвой схемы, реализующей $f$.

**?** Сколько сто́ит медиана $(m)$? $m(x_1, x_2, x_3) = x_1x_2 \oplus x_1x_3 \oplus x_2x_3$
Сколько сто́ит самая дорогая функция от $n$ переменных?

**Ответы и пояснения — на странице 3**

## Третья задача

Ниже приведены две попытки реализовать сортировку массива целых чисел.
(`swap(arr,i,j)` *меняет местами* `i`*-й и* `j`*-й элементы массива* `arr`, *а* `Int` *— тип, содержащий* **все** *целые числа*)

```
// глупая сортировка (?)
void sort(Int * arr, Int len) {
 for(Int i = 0; i < len; ++i) {
  if(arr[i] <= arr[i+1]) continue;
  swap(arr, i, i+1);
  i = -1;
}}
```

```
// сортировка пузырьком (?)
void sort(Int * arr, Int len) {
 for(Int i = 0; i < len; ++i)
  for(Int j = i; j < len; ++j)
   if(arr[i] > arr[j]) swap(arr,i,j);
}
```

**?** Преобразуют ли эти функции массив `arr` длины `len` одинаковым образом?
А это преобразование — действительно сортировка массива?
Попробуйте **строго обосновать** свои ответы.

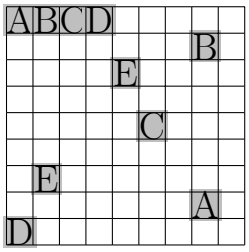
## Четвёртая задача

Рассмотрим поле (справа), составленное из квадратных клеток.

Требуется соединить каждую пару одинаковых букв *маршрутом*.

Через каждую клетку может проходить только один маршрут.

Маршрут может продолжаться в соседнюю клетку по вертикали и по горизонтали, но не по диагонали.




**?** Соедините все пары одинаковых букв маршрутами.
Какой стратегии следует придерживаться, чтобы **быстро** провести все маршруты в произвольном прямоугольнике для произвольной расстановки букв, если это возможно?

## Пятая задача

Даны $N$ квадратов со длинами сторон $1, 2, \ldots, N$ соответственно. Эти квадраты разрешено располагать на плоскости так, чтобы стороны были параллельны осям декартовых координат и внутренности не пересекались.

**?** Разместите **а)** 5, **б)** 10, **в)** 20, **г)** 25 квадратов так, чтобы ограничивающий их прямоугольник имел наименьшую возможную площадь.
Какой стратегии следует придерживаться, чтобы *быстро* разместить квадраты для любого $N$?

**Ответы и пояснения — на странице 4**

## Ответы и пояснения к задаче 1

Стоимость медианы — 1:

$$y_1 = x_1 \oplus x_2; \quad y_2 = x_1 \oplus x_3; \quad y_3 = y_1 \,\&\, y_2; \quad z = y_3 \oplus x_1$$

Этой задачей обозначено одно из направлений исследований на кафедре: изучение форм представления функций и решение алгоритмических задач, связанных с этими формами.

Функцию $f(n)$ стоимости самой дорогой функции от $n$ переменных принято называть **функцией Шеннона**. На кафедре исследуются такие функции для самых разных определений стоимости в самых разных дискретных задачах, и понять, как хотя бы приблизительно устроена эта функция, часто бывает непросто.

## Ответы и пояснения к задаче 2

Стоимость **любой** булевой функции — 1: достаточно реализовать функцию схемой, устроенной так же, как и полином Жегалкина, и для проверки исправности посмотреть на значение схемы на наборе $(1, 1, \ldots, 1)$. Решение выглядит простым, но додуматься до него не очень просто: это треть известной научной статьи 2000-го года.

Этой задачей обозначено одно из направлений исследований на кафедре: вопросы надёжности и контроля управляющих систем.

Функцию $f(n)$ стоимости самой дорогой функции от $n$ переменных принято называть **функцией Шеннона**. На кафедре исследуются такие функции для самых разных определений стоимости в самых разных дискретных задачах, и понять, как хотя бы приблизительно устроена эта функция, часто бывает непросто.

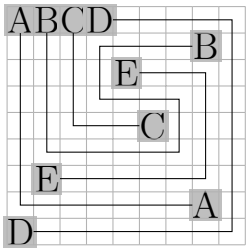
## Ответы и пояснения к задаче 3

`Сортировка пузырьком` действительно сортирует массив целых чисел.

А в `глупую сортировку` вкралась ошибка: вместо "`i < len`" должно быть "`i < len - 1`".

Первый вопрос касается *проблемы эквивалентности программ*: выяснить, делают ли программы одно и то же. Второй вопрос демонстрирует на простом примере, зачем нужна *формальная верификация программ*: строгая проверка соблюдения ими предъявляемых требований. А над тем, как задавать такие вопросы и получать на них **математически достоверные** ответы, размышляют на кафедре.

## Ответы и пояснения к задаче 4




Чтобы решить эту задачу, достаточно перебрать все варианты маршрутов.

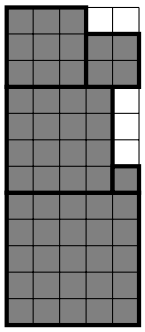
Придумав хорошую стратегию, можно заметно сократить перебор.

Но существование стратегии быстрого соединения произвольно расставленных букв маршрутами в произвольном прямоугольнике **крайне сомнительно**: *распознавательный аналог задачи (проверить, существует ли решение) является NP-полным.*

Эта задача — задача *глобальной трассировки* — относится к области анализа и синтеза сверхбольших интегральных схем, исследуемой на кафедре.

## Ответы и пояснения к задаче 5

Ответ для (а):



Немного подумав, можно решить эту задачу перебором конечного числа вариантов (последовательно "пристыковывая" квадраты друг к другу всеми способами).

Подумав ещё немного, можно заметно сократить перебор, придуманный "навскидку".

Но существование стратегии быстрого расположения квадратов на плоскости для произвольного $N$ **крайне сомнительно**: *распознавательный аналог задачи (для произвольных целых чисел $N$ и $k$ проверить, можно ли разместить $N$ квадратов в прямоугольнике площади $k$) является NP-полным.*

Эта задача — задача *размещения* — относится к области анализа и синтеза сверхбольших интегральных схем, исследуемой на кафедре.