Программирование+ + настольные игры с ИКИТом

Выпуск №9

Разработчик: Халтурин Е.

Алгоритм Евклида (527)

сложность

Ссылка

Алгоритм Евклида (527)

В данной задаче действительно реализуется стандартный алгоритм Евклида. Действительно, как соотнести переход **a** = **a** mod **b**, с переходом **a** = **a** –**b**?
Для этого рассмотрим шаг, на котором **b** = **d** (то есть одно из необходимых равенств выполнено). Теперь можно заметить, что если **a** mod **b** ≡ **c** mod **b**, то значит, при последовательном выполнении операции **a** = **a** –**b**, на каком-то из шагов мы получим в точности **c**.

Алгоритм Евклида (527)

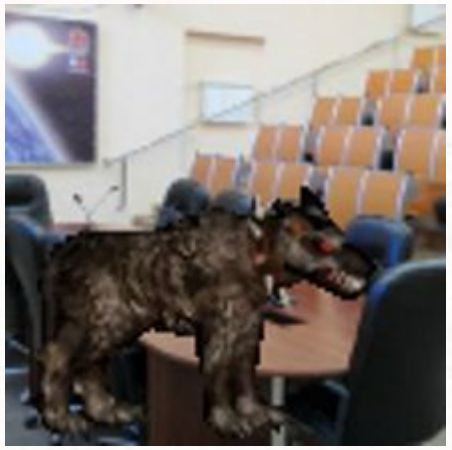
Доказательство:
Дано **a** mod **b** ≡ **c** mod **b** ≡ **r**. Требуется доказать, что **a** – **kb** = **c**.

a = **k1*b** + **r**; **c** = **k2*b + r.** Если **k2<k1**, то ∃ m, что **a** – **mb** = **c**. Верно, так как если **m** = **k1 – k2**, то **a** – **mb** = **k1*b** + **r** – **(k1-k2)*b** = **k2*b** + **r** = **c**, что и требовалось доказать.

Школа программиста

Баланс скобок (899)

сложность



Ссылка

Баланс скобок (899)

Как выявлять скобочную последовательность на наличие ошибок?

Для этого достаточно разобрать, какие ошибки могут быть:

1) Если для закрывающей скобки нет открывающей;
2) Если для закрывающей скобки первого вида ближайшей открывающей будет скобка другого вида;
3) Если останутся не закрытые открывающие скобки.

Баланс скобок (899)

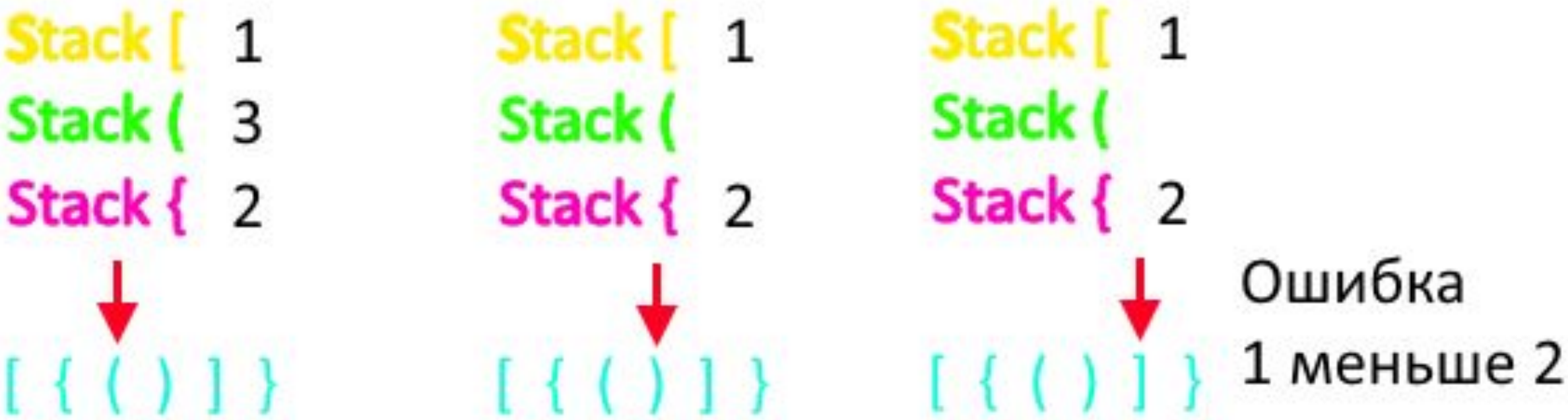
Будем рассматривать скобочную последовательность пошагово.
Для того, чтобы проверить первое условие, достаточно ввести переменную, указывающую на количество не закрытых скобок.
Также можно проверить и третье условие, если данная переменная, после всей проверки последовательности, не обнулена, значит условие не выполнено.

Баланс скобок (899)

Наибольший интерес представляет второй случай.
Для его проверки необходимо ввести несколько стеков, в которых будут запоминаться позиции открывающих скобок определённого типа.
Для проверки условия на шаге с закрывающей скобкой, достаточно проверить, что на вершине стека данного типа скобок число наибольшее, нежели на вершинах других стеков.

Баланс скобок (899)

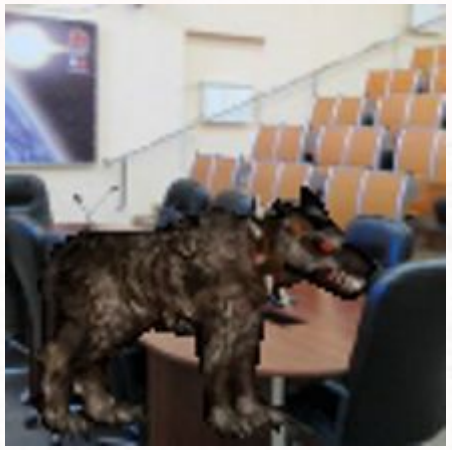
Пример: [{(}]}



Школа программиста

Точки и отрезки (396)

сложность



Ссылка

Точки и отрезки (396)

Воспользуемся сужение индексов, диапазон чисел из -10^9 до 10^9 можно сузить до количества уникальных элементов, то есть до $3*10^5$.
Для этого достаточно записать все числа в один единый массив и отсортировать его, полученными индексами можно заменить исходное число, не меняя ответа на задачу.

Точки и отрезки (396)

Пример:	Полученные данные:
3 2	3 2
0 5	2 5
-30 2	1 4
70 100	7 8
1 6	3 6

То есть получится единый массив:

Индекс	1	2	3	4	5	6	7	8
Число	-30	0	1	2	5	6	70	100

Точки и отрезки (396)

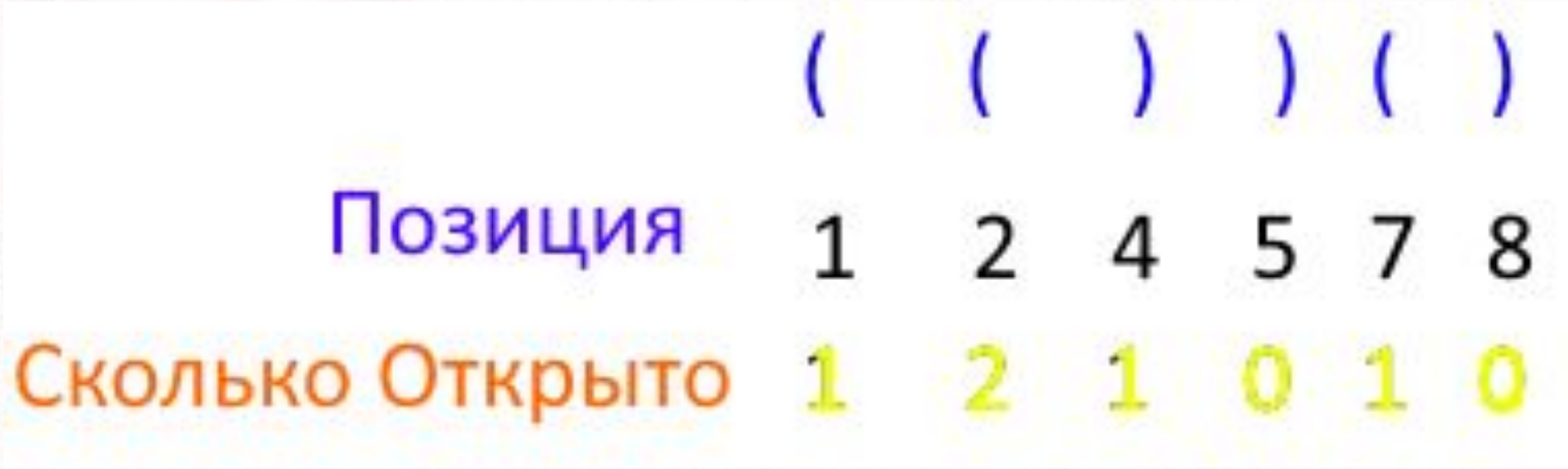
Чтобы не расходовать дополнительную память, индекс соответствия можно находить бинарным поиском.

Создадим массив, в котором для каждой позиции, на которой находится скобка, будет отмечено количество открытых в данный момент скобок. Позиции закрывающих скобок сдвинем на один вправо, чтобы узнать позицию, на которой количество открытых скобок уменьшилось.

Точки и отрезки (396)

Пример:

3 2
2 5
1 4
7 8
3 6



Точки и отрезки (396)

Далее, для каждого запроса бинарным поиском ищем ближайшую слева скобку (её позицию). В качестве ответа выдаём количество скобок, открытых на найденной позиции.

Адаптивный поиск (647)

сложность

Ссылка

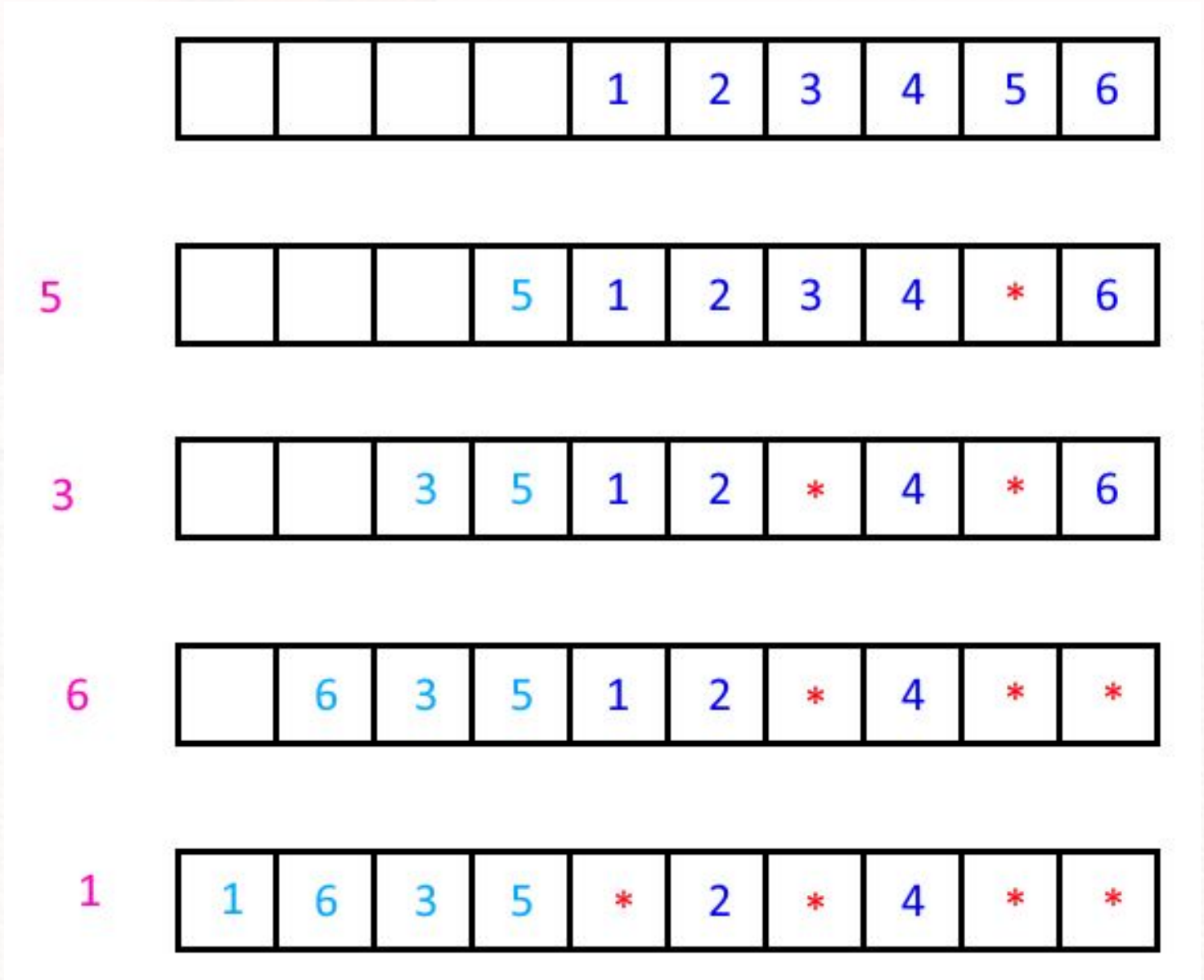
Адаптивный поиск (647)

Как быстро осуществлять перестановку числа в начало.

Можно использовать хитрость. Пусть массив, в котором хранятся числа, вначале будет иметь пустое место. Тогда нам достаточно записать переставляемое число в ту позицию, которая является ближайшей незаполненной.

Адаптивный поиск (647)

Пример:
6 4
5 3 6 1



Адаптивный поиск (647)

Теперь, для точного нахождения позиции числа, необходимо быстро находить количество пропусков (красных звездочек) до текущей позиции числа.
Для этого, можно использовать любую структуру быстрого подсчёта: корневую декомпозицию, дерево Фенвика, дерево отрезков.

Школа программиста

Трамвай (532)

сложность

Ссылка

Трамвай (532)

Будем решать задачу постепенно, с шагом в одну остановку.
Заведём две переменные, отвечающие за сумму настроения всех тех, кто стоит, и тех, кто сидит.
Как же понять, кто должен сидеть?
Для этого заведём кучу [priority_queue], хранящую эффективность как ключ (разность настроения между тем, как сидеть и стоять) и номер человека. На вершине будет человек с минимальной эффективностью.

Трамвай (532)

Когда приходит новый человек, если его эффективность положительная и места ещё есть, он садится, если его эффективность больше того человека, кто на вершине кучи, он тоже садится, иначе едет стоя.

Когда кто-то выходит, кто-то из стоящих может занять освободившееся место. Для этого заведём ещё одну кучу [priority_queue], для стоящих, с человеком, у которого максимальная эффективность, на вершине.

Трамвай (532)

Чтобы каждый раз не пересчитывать сумму всей кучи стоящих и кучи сидящих, и не использовать более сложные структуры данных, можно обойтись переменной, хранящая сумму.

Чтобы не удалять человека, который выходит из трамвая, из середины кучи, можно делать ленивое удаление. Для этого будем запоминать состояние каждого человека (стоит, сидит, ушёл), и пока на вершине будет ушедший человек, будем производить удаление.

Для ленивого удаления понадобится переменная, хранящая действительный размер кучи.

Трамвай (532)

Пример:

5 2 5

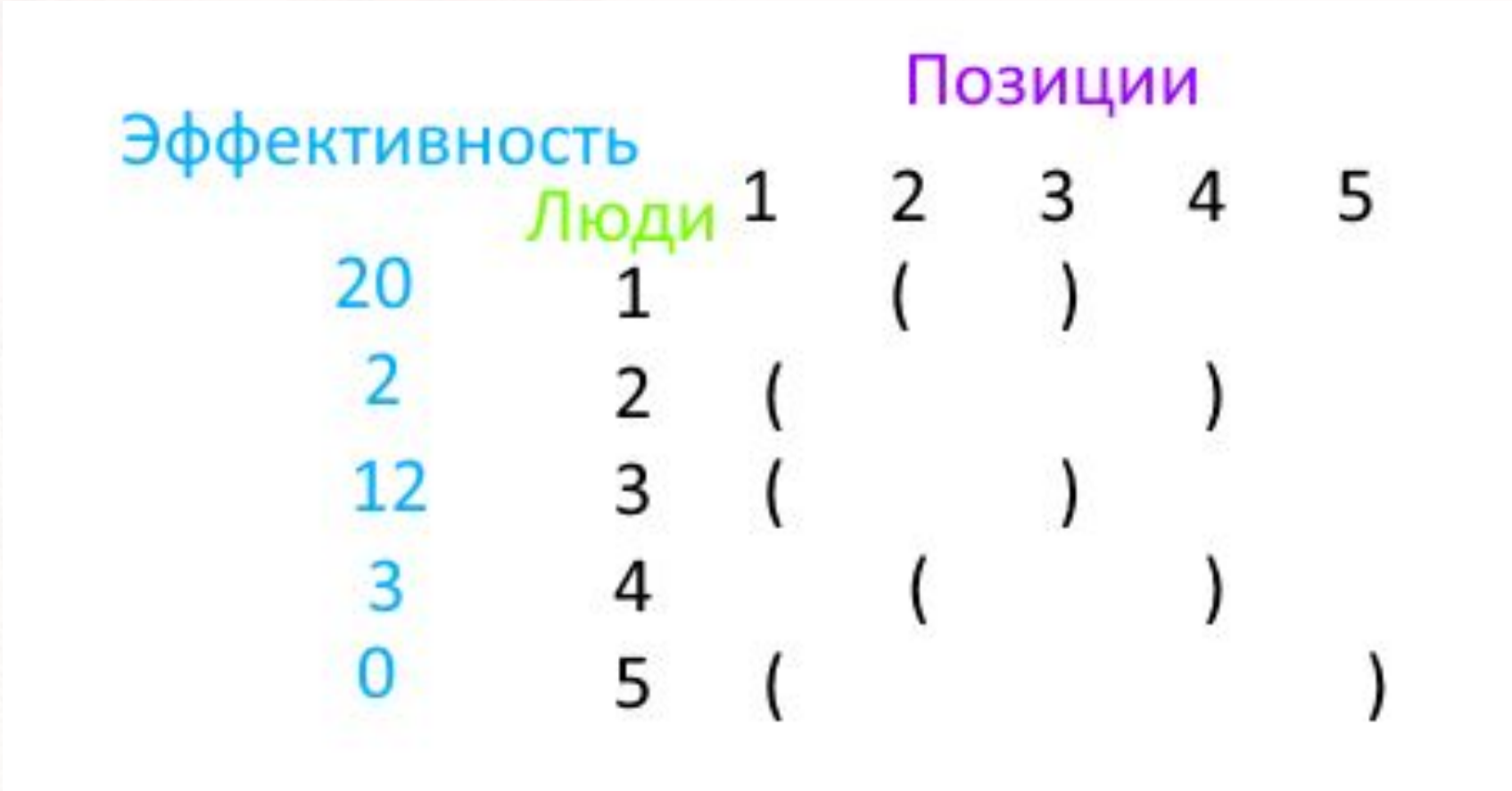
10 -10 2 3

-1 -3 1 4

6 -6 1 3

7 4 2 4

4 4 1 5



Трамвай (532)

Пример:

5 2 5

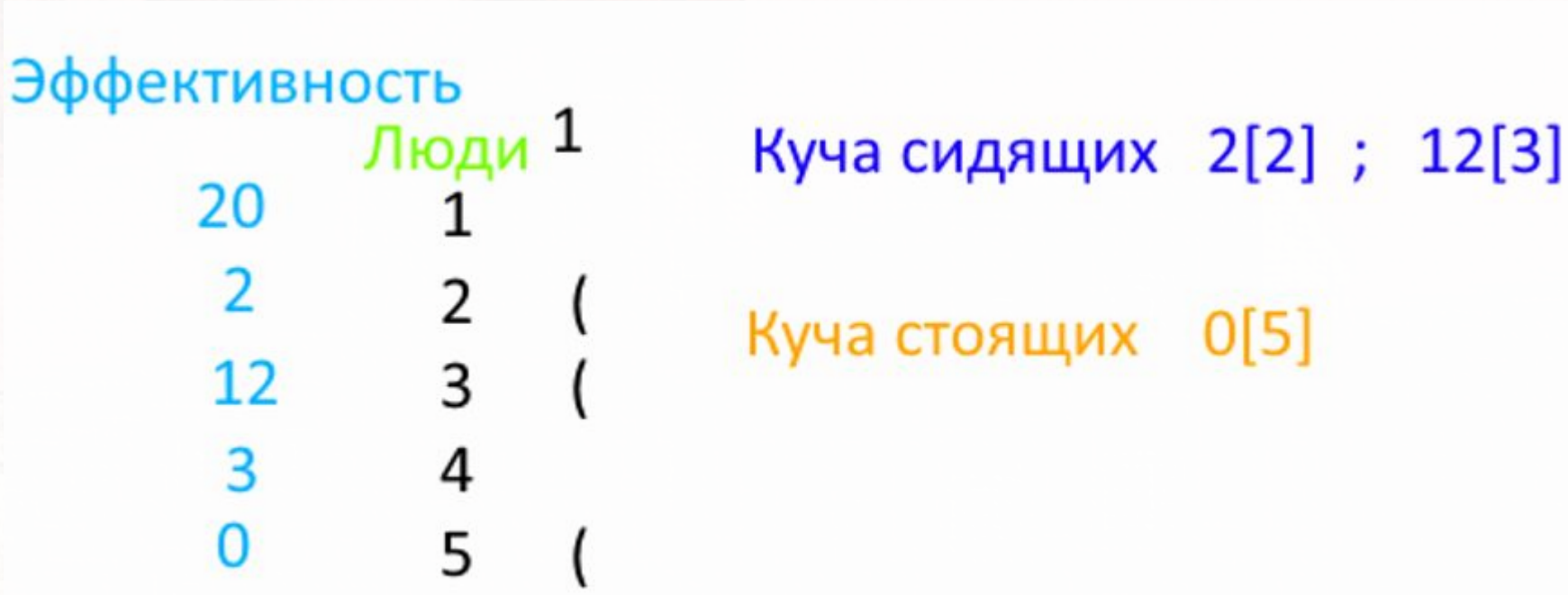
10 -10 2 3

-1 -3 1 4

6 -6 1 3

7 4 2 4

4 4 1 5



Трамвай (532)

Пример:

5 2 5

10 -10 2 3

-1 -3 1 4

6 -6 1 3

7 4 2 4

4 4 1 5

Трамвай (532)

Пример:
5 2 5
10 -10 2 3
-1 -3 1 4
6 -6 1 3
7 4 2 4
4 4 1 5

Программирование+ + настольные игры с ИКИТом

Текст задач взят с сайтов:

- acmp.ru
- codeforces.com
- informatics.mccme.ru

Выпуск №9

Разработчик: Халтурин Е.