2.2.4 Найти максимальное произведение длины подотрезка на минимум на нем.

Для начала добавим в начало и конец нашего массива два нуля, так будет проще.
Потом найдем для каждого числа ближайшее справа меньшее его.
Затем для каждого же числа – ближайшее слева меньшее его.
Таким образом, для каждого элемента определяется отрезок, на котором **он** является минимальным. Перебрав все числа, умножив длину соответствующего отрезка на это (текущее) число и выбрав максимум, получим ответ.

Теперь детали: как выбирать ближайший минимальный справа. (Слева аналогично, но с конца)
Заведем стек, положим туда первый ноль. Далее будем заполнять стек двигаясь по числу слева направо. Как только вершина стека окажется **больше** чем вновь запихиваемое число, сделаем поп и тому, что только что ушло из стека поставим ссылку на новое число – это будет его ближайшее меньшее справа. Если вершина до сих пор больше нового числа – то и его выкинем из стека и ему пропишем ссылку на новое число. Итак пока вершина не станет меньше либо равна нового числа (в крайнем случае опустимся до нуля – стек не опустеет, так как по условию у нас все числа натуральные). В конце положим-таки новое число в стек. И так далее.
В конце, когда дойдем до последнего нуля – он убьет весь стек и все числа, что были в стеке получат ссылку на этот ноль. Это будет значить, что у этих чисел не было справа от них меньших чисел.

Ссылки будем хранить в виде второго массива, скажем B. B[i] будет индексом в массиве A, который равен индексу ближайшего меньшего справа числа A[i].

Линейность следует из наблюдения о том, что каждое число ровно один раз помещается в стек и ровно один раз из него вынимается.

2.2.5*. Найти в массиве объектов, над которыми задана операция сравнения, элемент, встречающийся больше «половины» раз.

Будем хранить набор одинаковых элементов, но не в виде массива, а в виде указателя на элемент и их количества – назовем их кандидаты на ответ.
Пробежимся по массиву (индексом или итератором – неважно). Если в рассматриваемый момент число кандидатов равно нулю, то сделаем текущий элемент кандидатом и количество установим в 1. Если же их не ноль (то есть существует ненулевое число одинаковых элементов – в том числе и один), то сравним с текущим элементом. Если они равны – увеличиваем число кандидатов, если нет – уменьшаем. Алгоритм закончен. Ответом будет наш кандидат. Всего один проход по массиву.

Почему он работает: фактически мы «разбиваем» наш массив на пары «различающихся» элементов – и если в конце у нас есть ненулевое число «кандидатов» - то это и будет нашим ответом, ибо искомых элементов строго больше чем число пар. Потом хорошо бы проверить, что этот элемент действительно наш искомый, то есть требуется второй проход, но раз **гарантируется**, что такой элемент есть – то второй проход и не нужен.