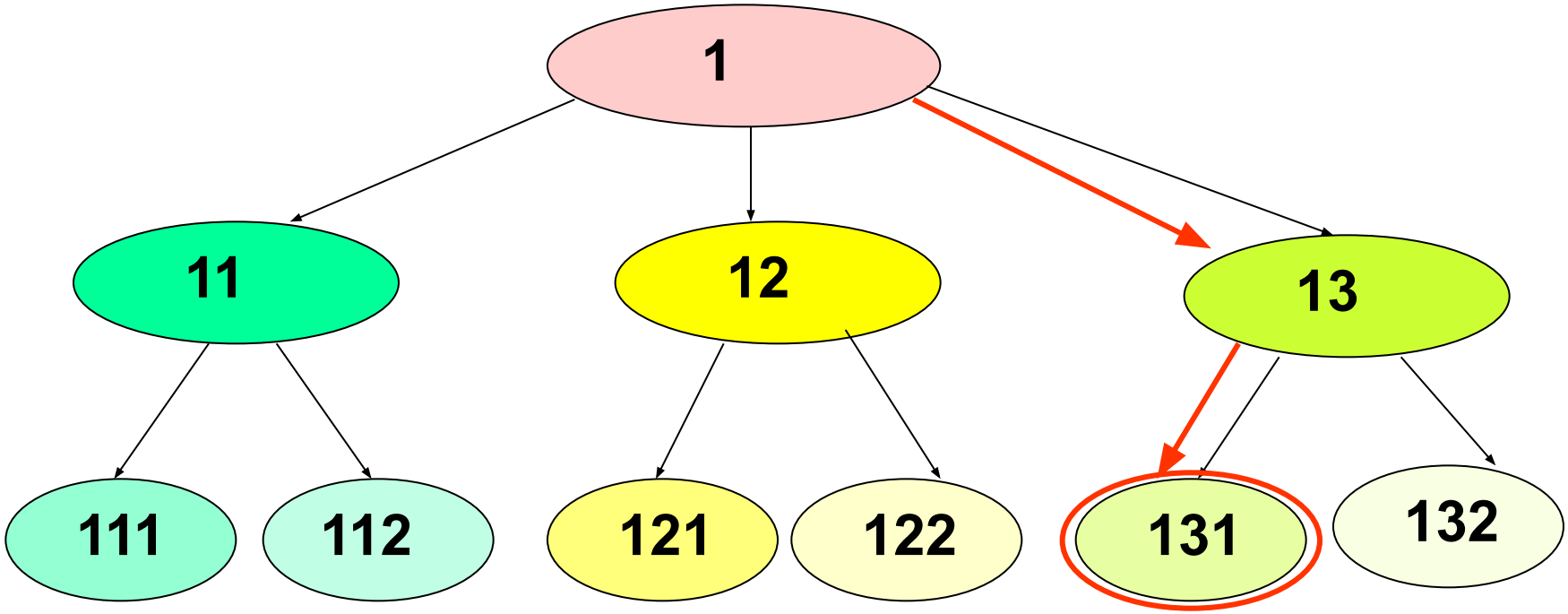
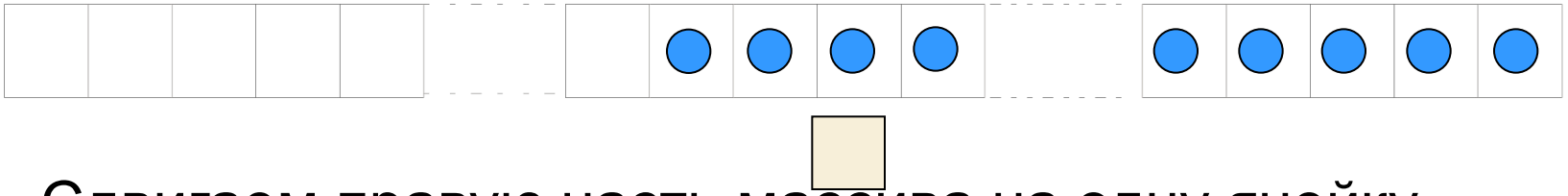
Большие списки

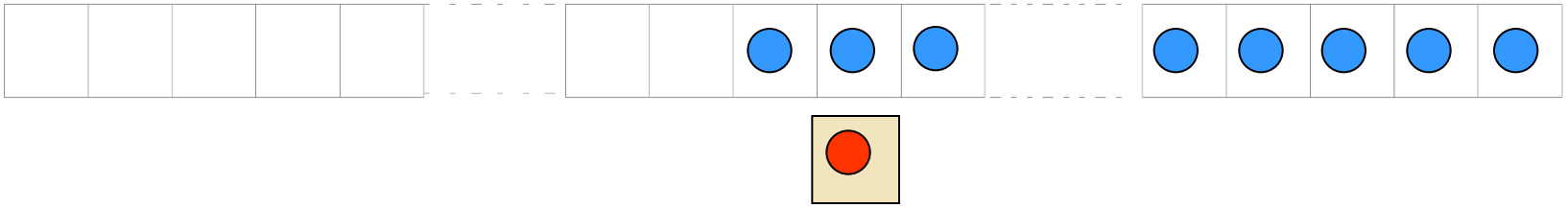


- Очень большие списки можно разбить на подсписки и выбрать один подсписок, в котором содержатся интересующие нас элементы.

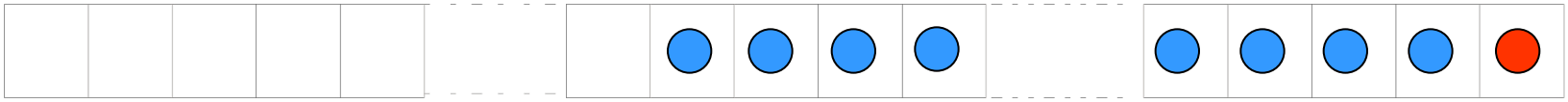
- Находим в массиве наибольший элемент и помещаем его в буфер



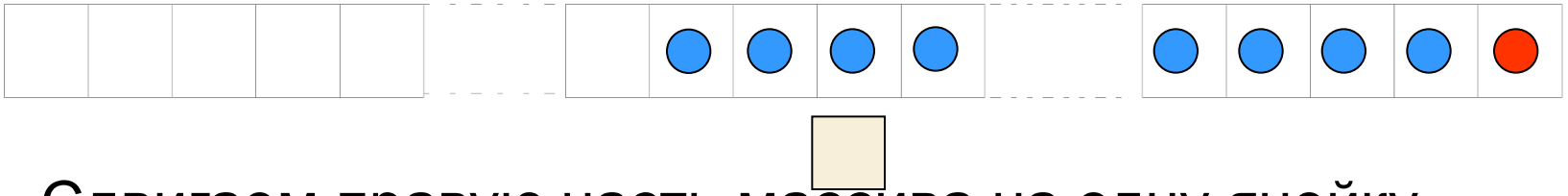
- Сдвигаем правую часть массива на одну ячейку влево



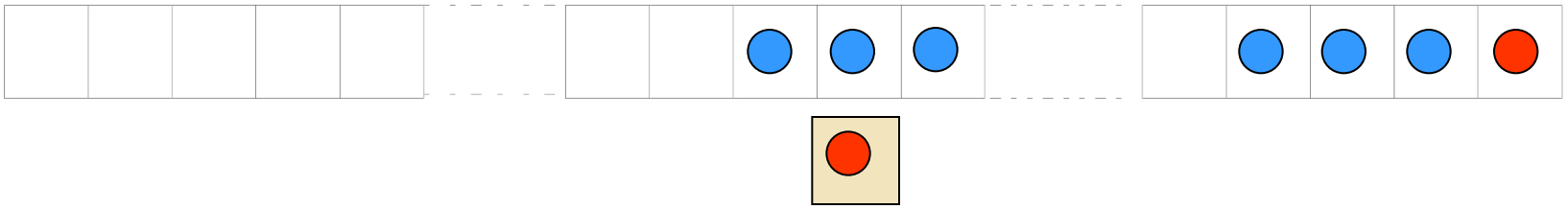
- Наибольший элемент из буфера перемещаем в конец массива



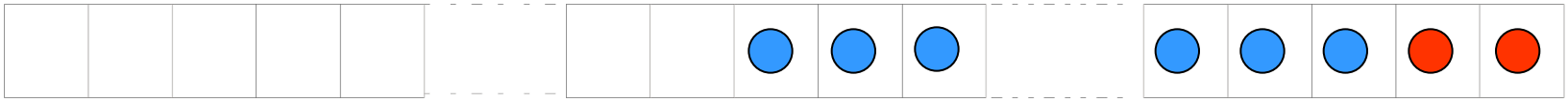
- Находим в массиве второй по величине элемент и помещаем его в буфер



- Сдвигаем правую часть массива на одну ячейку влево (кроме последнего элемента)

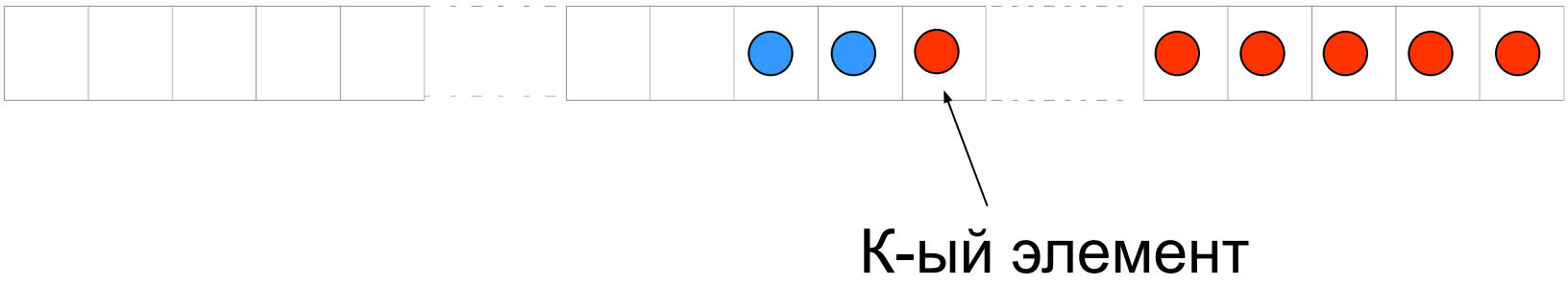


- Второй по величине элемент из буфера перемещаем в конец массива на освободившееся место

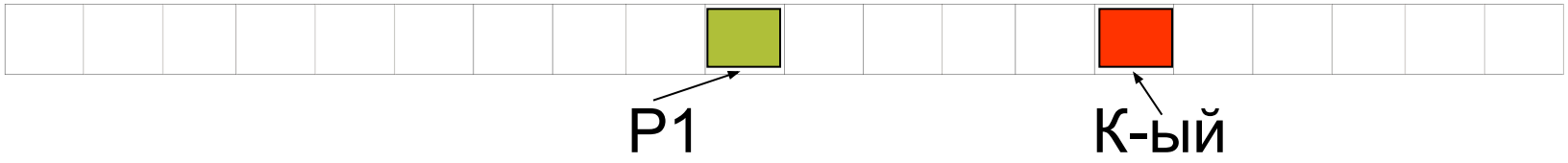


К-ый элемент

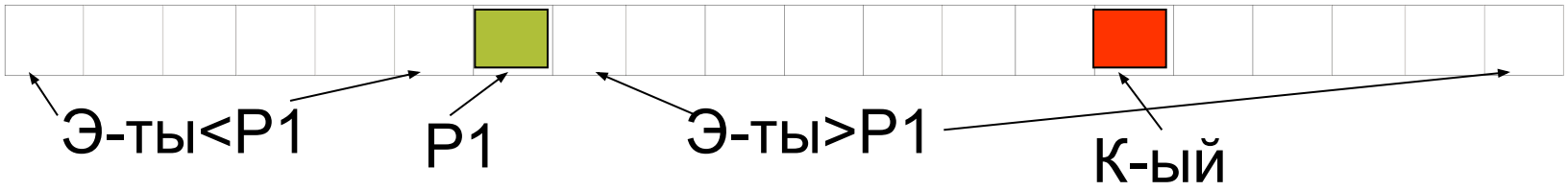
- Далее процесс повторяется до расположения ***к-***го по величине элемента на нужном месте



- В массиве задаем опорный элемент “Р1”

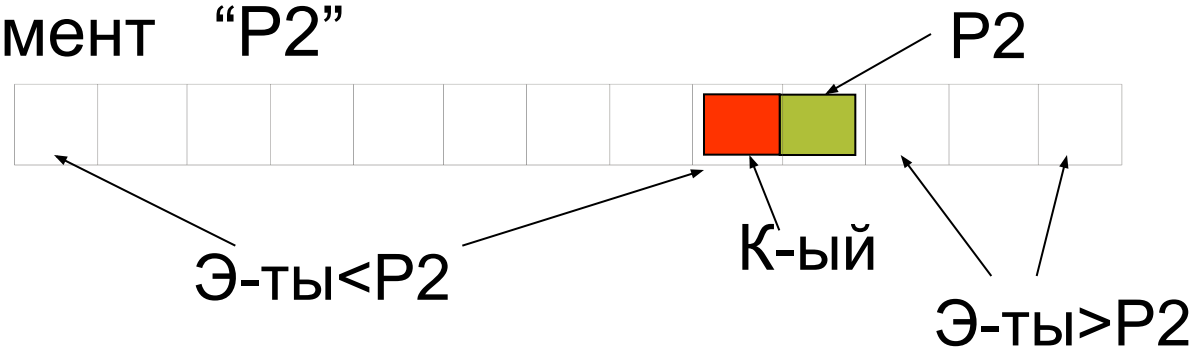


- Просматриваем массив слева направо, располагая элементы меньшие Р1 слева от него, а большие элементы - справа от него



- Исключаем из рассмотрения левую часть массива

- В оставшейся части массива задаем опорный элемент “P2”



- Просматриваем массив слева направо, располагая элементы меньшие P2 слева от него, а большие элементы - справа от него

- Исключаем из рассмотрения правую часть массива

- Продолжаем этот процесс.
- Возможны два варианта завершения:

1. На очередной итерации опорный элемент ***Pi*** располагается на ***K***-ой позиции.
2. Процесс сокращения интервала идет до логического конца – остается подинтервал из двух элементов, один из которых располагается на ***K***-ой позиции.

Алгоритмы сортировки

- Есть последовательность a_1 , a_2... a_n и функция сравнения, которая на любых двух элементах последовательности принимает одно из трех значений: меньше, больше или равно.
- Задача сортировки состоит в перестановке членов последовательности таким образом, чтобы выполнялось условие:

$$a_i \leq a_{i+1}$$ для всех ***i*** от ***0*** до ***n***.

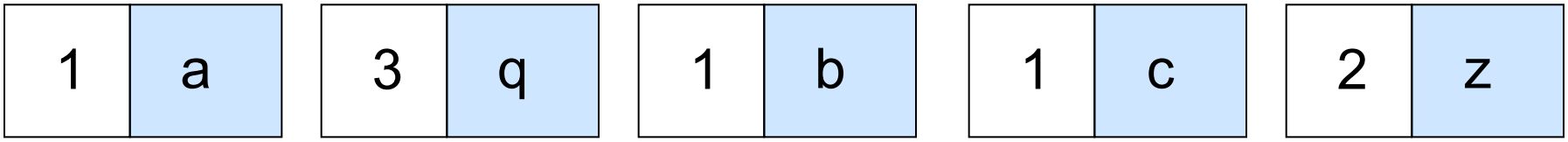
- Возможна ситуация, когда элементы состоят из нескольких полей:

- Если значение функции сравнения зависит только от поля ***x*** , то ***x*** называют ключом, по которому производится сортировка.
- На практике, в качестве ***x*** часто выступает число, а поле ***y*** хранит какие-либо данные, никак не влияющие на работу алгоритма.

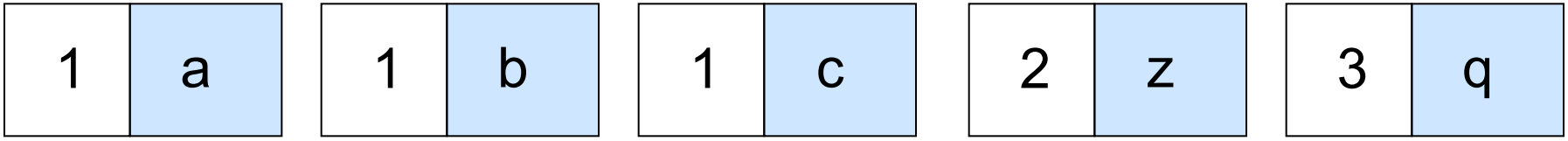
Критерии эффективности работы алгоритмов

1. ***Время сортировки*** - основной параметр, характеризующий быстродействие алгоритма.
2. ***Память*** - ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных. При оценке используемой памяти не учитывается место, которое занимает исходный массив и независящие от входной последовательности затраты, например, на хранение кода программы.

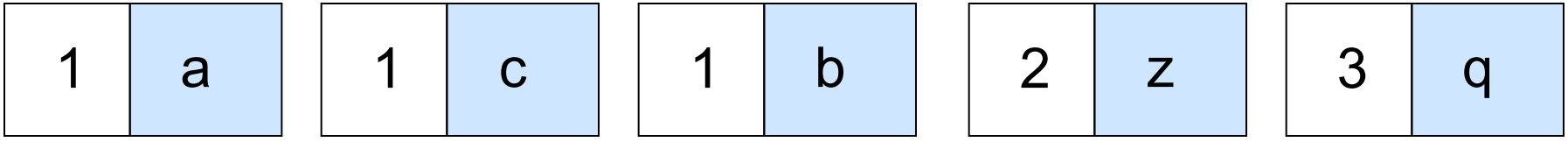
3. ***Устойчивость*** - устойчивая сортировка не меняет взаимного расположения равных элементов.



Расположение дополнительных полей "a", "b", "c" осталось прежним



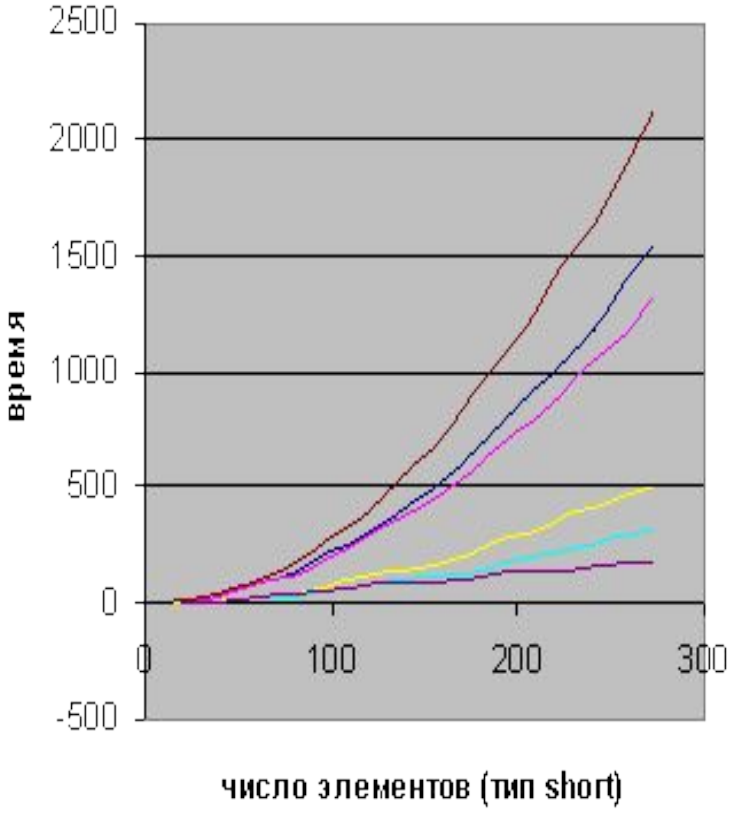
Взаимное расположение равных элементов с ключом 1 и дополнительными полями "a", "b", "c" изменилось



4. ***Естественность поведения*** - эффективность метода при обработке уже отсортированных, или частично отсортированных данных.

   Алгоритм ведет себя естественно, если учитывает эту характеристику входной последовательности.

Сравнение времени сортировок

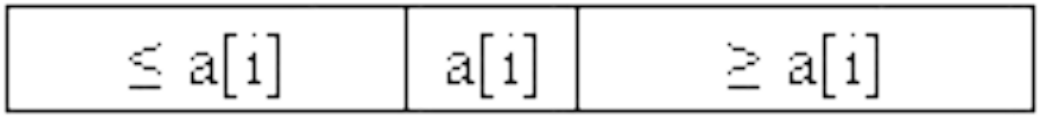


коричневая линия: сортировка пузырьком;
синяя линия: шейкер-сортировка;
розовая линия: сортировка выбором;
желтая линия: сортировка вставками;
голубая линия: сортировка вставками со сторожевым элементом;
фиолетовая линия: сортировка Шелла.

Быстрая сортировка

Метод основан на подходе "разделяй-и-властвуй". Общая схема такова:

- из массива выбирается некоторый опорный элемент a[i],
- запускается процедура разделения массива, которая перемещает все ключи, меньшие, либо равные a[i], влево от него, а все ключи, большие, либо равные a[i] - вправо,
- теперь массив состоит из двух подмножеств, причем левое меньше (либо равно) правого,

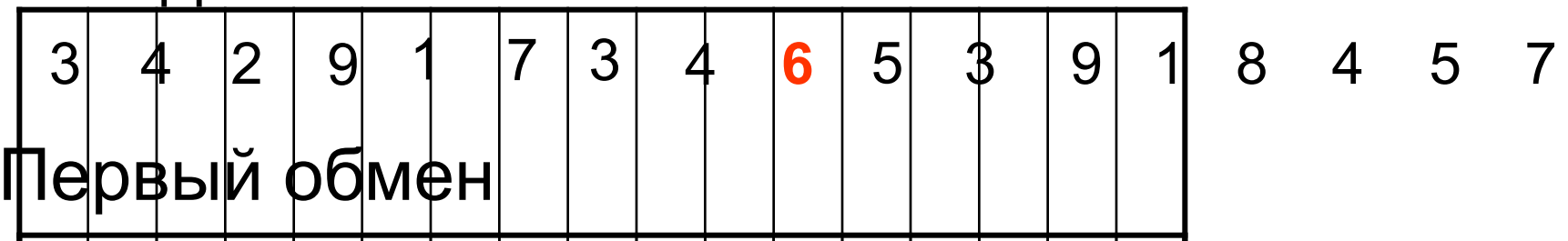


- для обоих подмассивов: если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру.

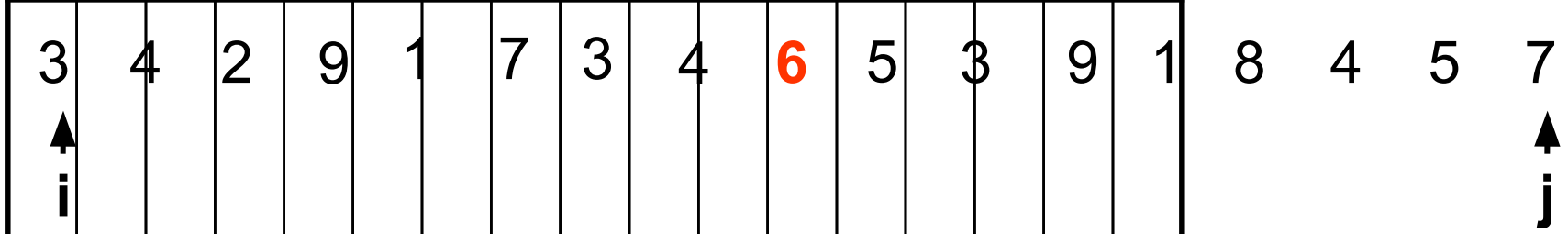
Разделение массива

- На входе массив a[0]...a[N] и опорный элемент p, по которому будет производиться разделение.
- Введем два указателя: i и j. В начале алгоритма они указывают, соответственно, на левый и правый конец последовательности.
- Будем двигать указатель i с шагом в 1 элемент по направлению к концу массива, пока не будет найден элемент a[i] >= p. Затем аналогичным образом начнем двигать указатель j от конца массива к началу, пока не будет найден a[j] <= p.
- Далее, если i <= j, меняем a[i] и a[j] местами и продолжаем двигать i, j по тем же правилам...
- Повторяем шаг 3, пока i <= j.

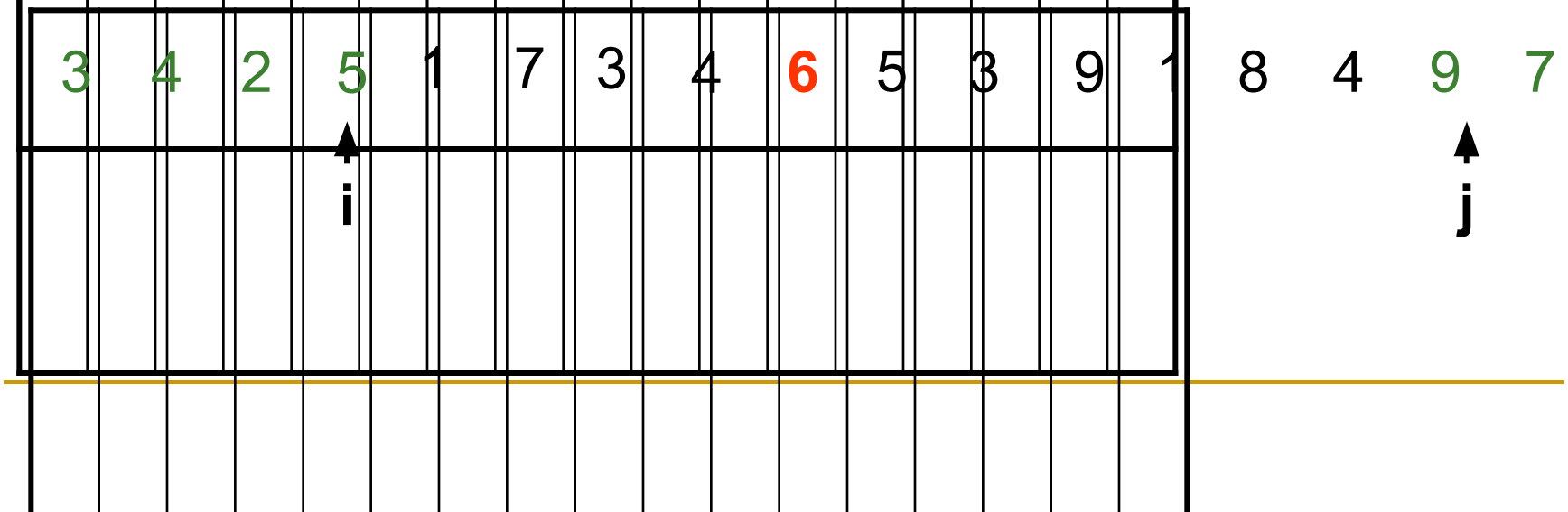
- Исходный массив



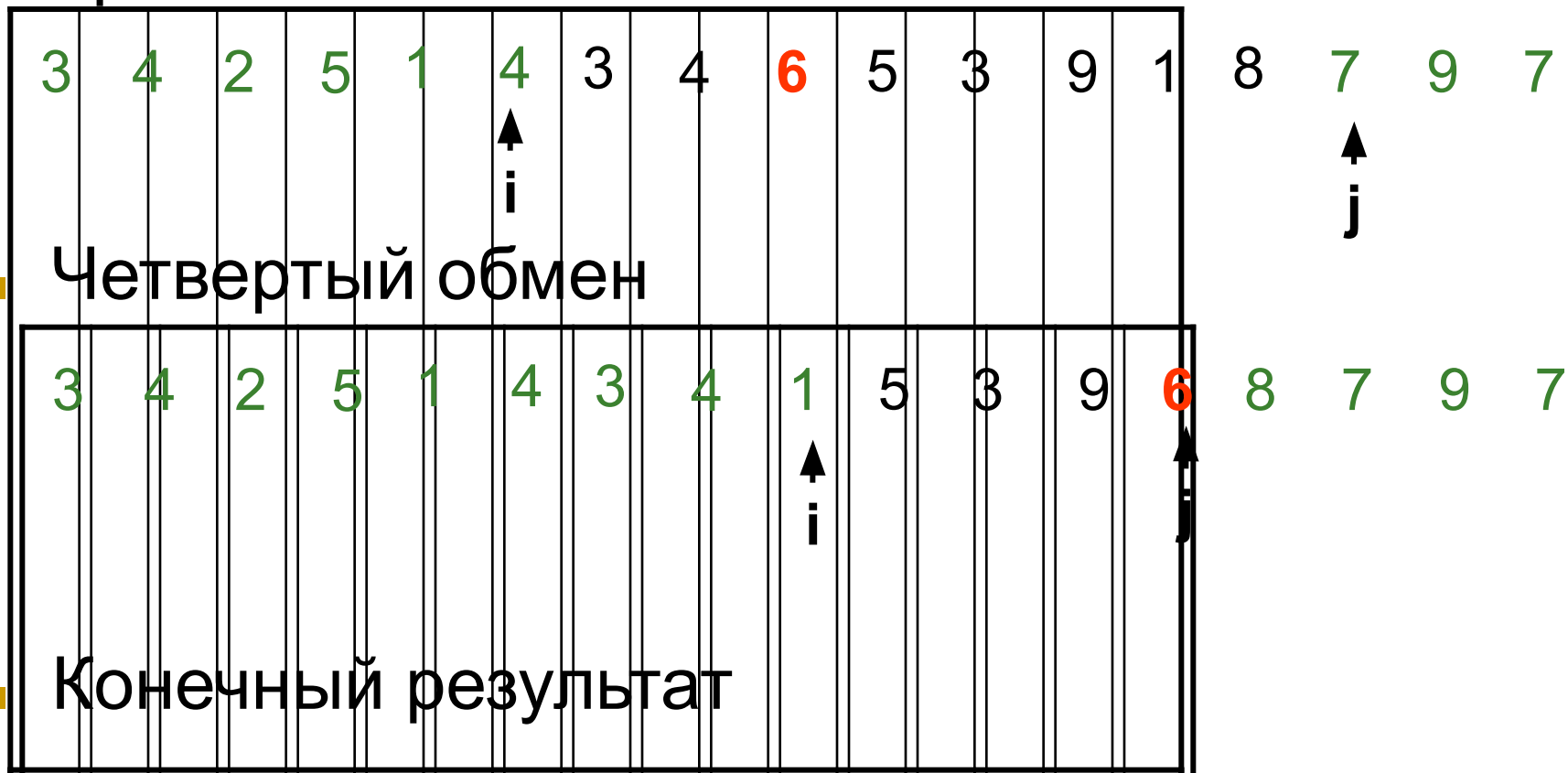
- Первый обмен



- Второй обмен

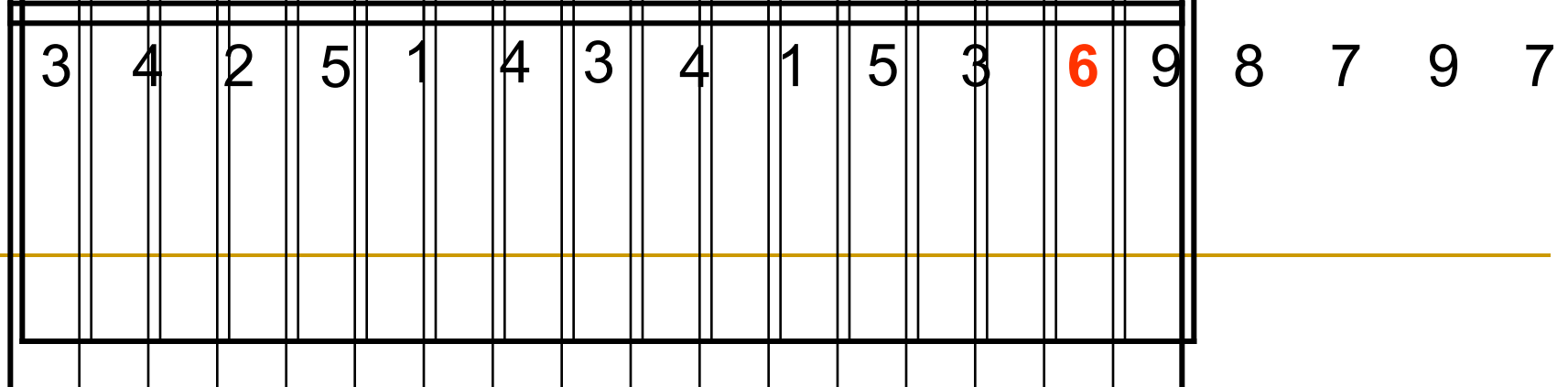


- Третий обмен



- Четвертый обмен

- Конечный результат



- Конечный результат

- Разделённые массивы

- Для разделённых массивов процедура повторяется

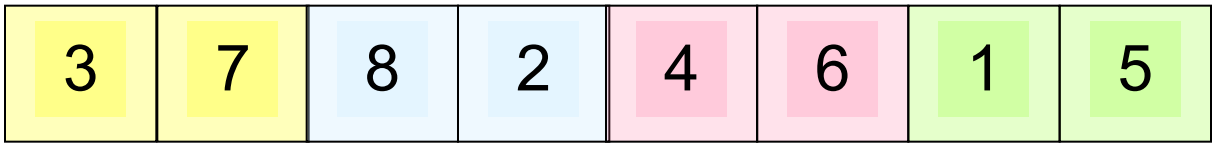
- ***Общее быстродействие метода -*** *хорошее.*
- ***Метод неустойчив****.*
- ***Поведение довольно естественно***, если учесть, что при частичной упорядоченности повышаются шансы разделения массива на более равные части.
- ***Сортировка использует дополнительную память***, так как данные о рекурсивных подвызовах каждый раз запоминать.

Сортировка слиянием

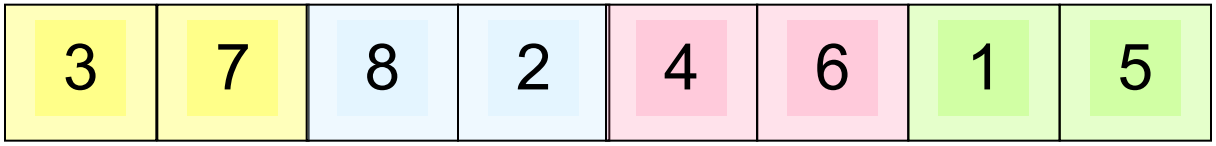
- Сортировка слиянием также построена на принципе "разделяй-и-властвуй", однако реализует его несколько по-другому, нежели быстрая сортировка.
- Вместо разделения по опорному элементу массив просто делится пополам.

- Рекурсивный алгоритм обходит получившееся дерево слияния в прямом порядке. Каждый уровень представляет собой *проход* сортировки слияния - операцию, полностью переписывающую массив.
- Деление происходит до массива из единственного элемента. Такой массив можно считать упорядоченным.
- Один из способов состоит в слиянии двух упорядоченных последовательностей при помощи вспомогательного буфера, равного по размеру общему количеству имеющихся в них элементов. Элементы последовательностей будут перемещаться в этот буфер по одному за шаг.

- Исходный массив

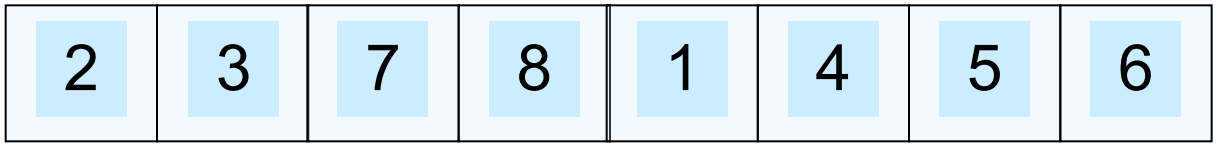


- Первый проход – пары

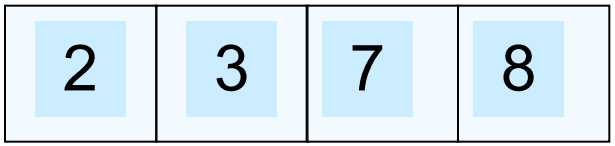


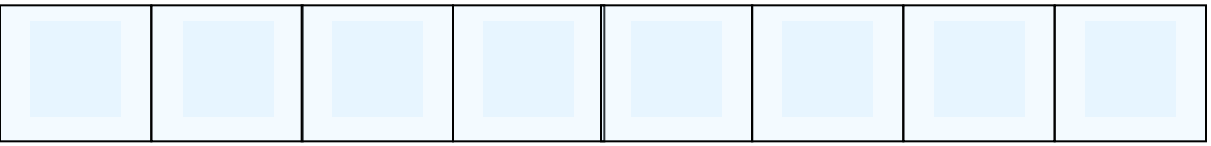
- Второй проход – четвёрки

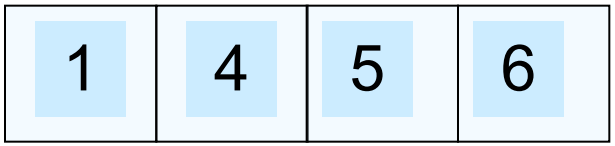
- Третий проход – восьмёрки



- Процесс слияния в буфер







- Сортировку слиянием используют для упорядочения массивов, лишь если требуется **устойчивость** метода (которой нет у быстрой сортировки).
- Сортировка слиянием является одним из наиболее эффективных методов для списков, когда есть лишь **последовательный доступ** к элементам.
- Несмотря на хорошее общее быстродействие и учитывая отсутствие "худшего случая", у сортировки слиянием есть и серьезный минус: она требует **дополнительную память**.