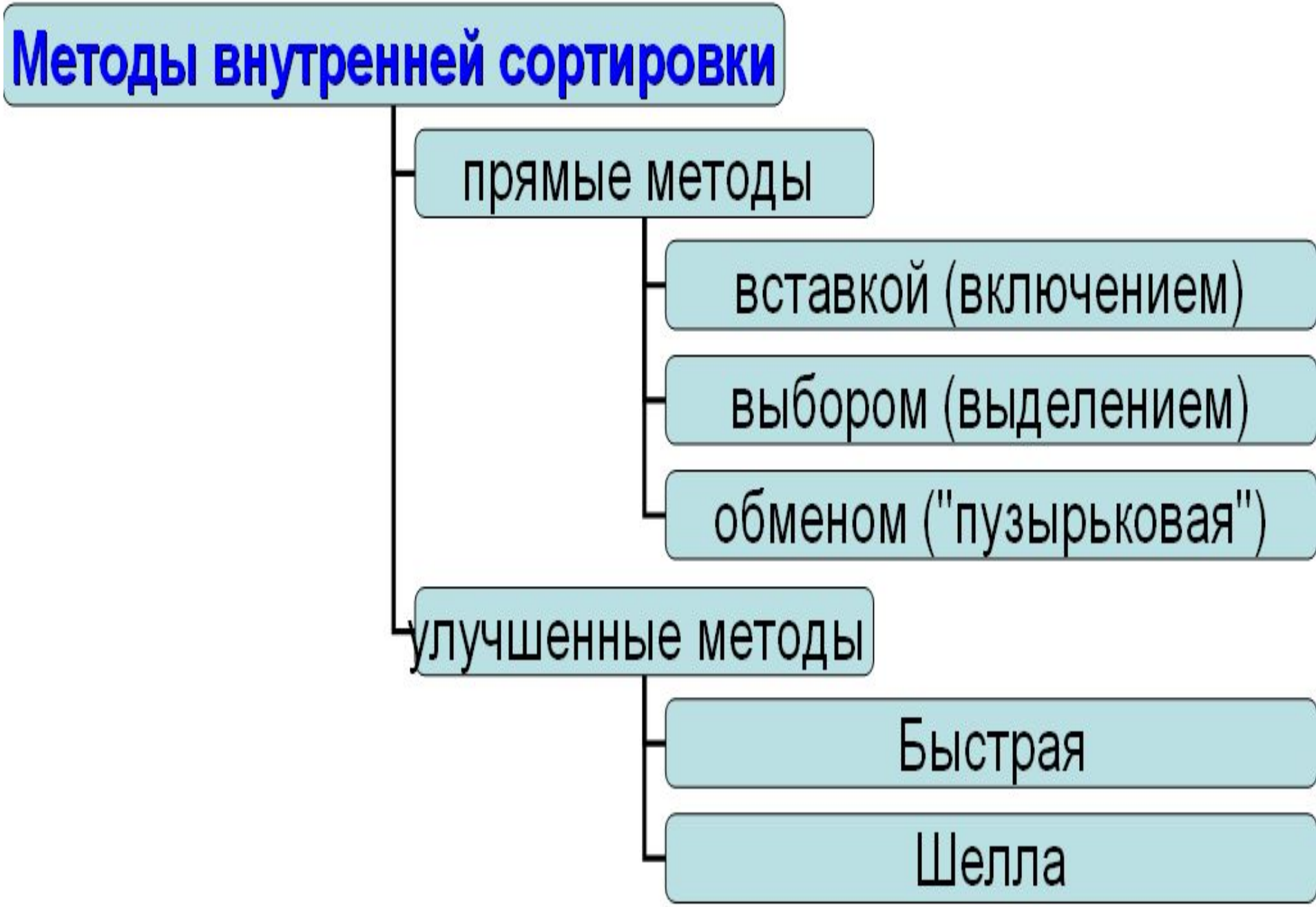
Методы сортировки данных


Методы внутренней сортировки
прямые методы
вставкой (включением)
выбором (выделением)
обменом ("пузырьковая")
улучшенные методы
Быстрая
Шелла

Алгоритм сортировки вставкой

Суть сортировки:

- ✔ Упорядочиваются два элемента массива
- ✔ Вставка третьего элемента в соответствующее место по отношению к первым двум элементам.
- ✔ Этот процесс повторяется до тех пор, пока все элементы не будут упорядочены.

Сортировка вставкой

по возрастанию

Массив отсортирован по возрастанию

Постановка задачи

Пусть нужно отсортировать массив **A** по возрастанию, в котором **N** элементов методом вставки

Вспомогательные переменные

j – номер первого элемента остатка.

i – номер перемещаемого элемента.

f – условие выхода из цикла (если **f=1**, то выход)

Val – промежуточное значение, используемое для перемещения элементов массив

Начало алгоритма.

Шаг 1 j:=2,

Шаг 2 **Пока** j<=N **выполнять**:

Шаг 2.1 i:=j; f:=0,

Шаг 2.2 **Пока** i>=2 **и** f=0 **выполнять**:

Шаг 2.2.1 **Если** A[i-1]>A[i]

то Val:=A[i-1];

A[i-1]:=A[i];

A[i]:=Val,

иначе f:=1,

Шаг 2.2.2 i:=i-1,

Шаг 2.3 j:=j+1.

Конец алгоритма.

Алгоритм сортировки выбором

Суть сортировки:

- Выбирается элемент с наименьшим значением и делается его обмен с первым элементом массива.
- Затем находится элемент с наименьшим значением из оставшихся n-1 элементов и делается его обмен со вторым элементом и т.д. до обмена двух последних

Сортировка выбором
по возрастанию

Постановка задачи

Пусть нужно отсортировать массив **A** по возрастанию, в котором **N** элементов методом выбора.

Вспомогательные переменные

j – номер первого элемента остатка.

i – номер перемещаемого элемента.

min – минимальное число в массиве.

Imin – номер минимального числа в массиве

Начало алгоритма.

Шаг 1 j:=1,

Шаг 2 Пока j<=N-1 **выполнять**:

Шаг 2.1 min:=a[j], Imin:=j, i:=j+1

Шаг 2.2 Пока i<=N **выполнять**:

Шаг 2.2.1 Если A[i]<min,

то min:=a[i], Imin:=i

Шаг 2.2.2 i:=i+1,

Шаг 2.3 A[Imin]:=A[j], A[j]:=min

Шаг 2.4 j:=j+1.

Конец алгоритма.

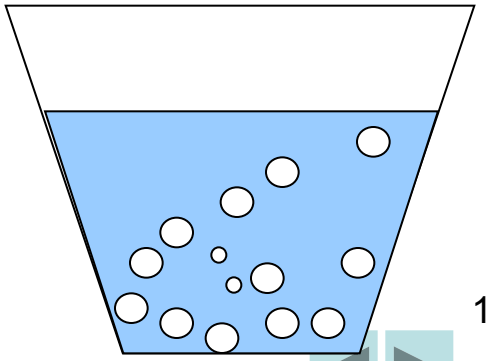
Алгоритм сортировки обменом («пузырьковая»)

Суть сортировки:

Последовательно просматривается массив и сравнивается каждая пара элементов между собой.

При этом "неправильное" расположение элементов устраняется путем их перестановки.

Процесс просмотра и сравнения элементов повторяется до просмотра всего массива.



Сортировка обменом

по возрастанию

Массив отсортирован по возрастанию

Постановка задачи

Пусть нужно отсортировать массив **A** по возрастанию, в котором **N** элементов методом обмена

Вспомогательные переменные

j – номер первого элемента остатка.

i – номер перемещаемого элемента.

Val – промежуточное значение, используемое для перемещения элементов массива

Начало алгоритма.

Шаг 1 j:=N,

Шаг 2 Пока j>=2 **выполнять**:

Шаг 2.1 i:=1; ,

Шаг 2.2 Пока i<=j-1**выполнять**:

Шаг 2.2.1 Если A[i]>A[i+1]

то Val:=A[i];

A[i]:=A[i+1];

A[i+1]:=Val,

Шаг 2.2.2 i=i+1,

Шаг 2.3 j:=j-1.

Конец алгоритма.

Сравнение соседних элементов

Обмен соседних элементов местами, в случае если левый больше правого

Формируется отсортированная часть

Алгоритм сортировки Шелла

- ✔ Классифицируется как «слияние вставкой»;
- ✔ Называется «сортировкой с убывающим шагом»
- ✔ Общий метод, который использует сортировку вставкой, применяет принцип уменьшения расстояния между сравниваемыми элементами

Условия реализации:

? Конкретная последовательность шагов может быть другой, но последний шаг должен быть равен 1;

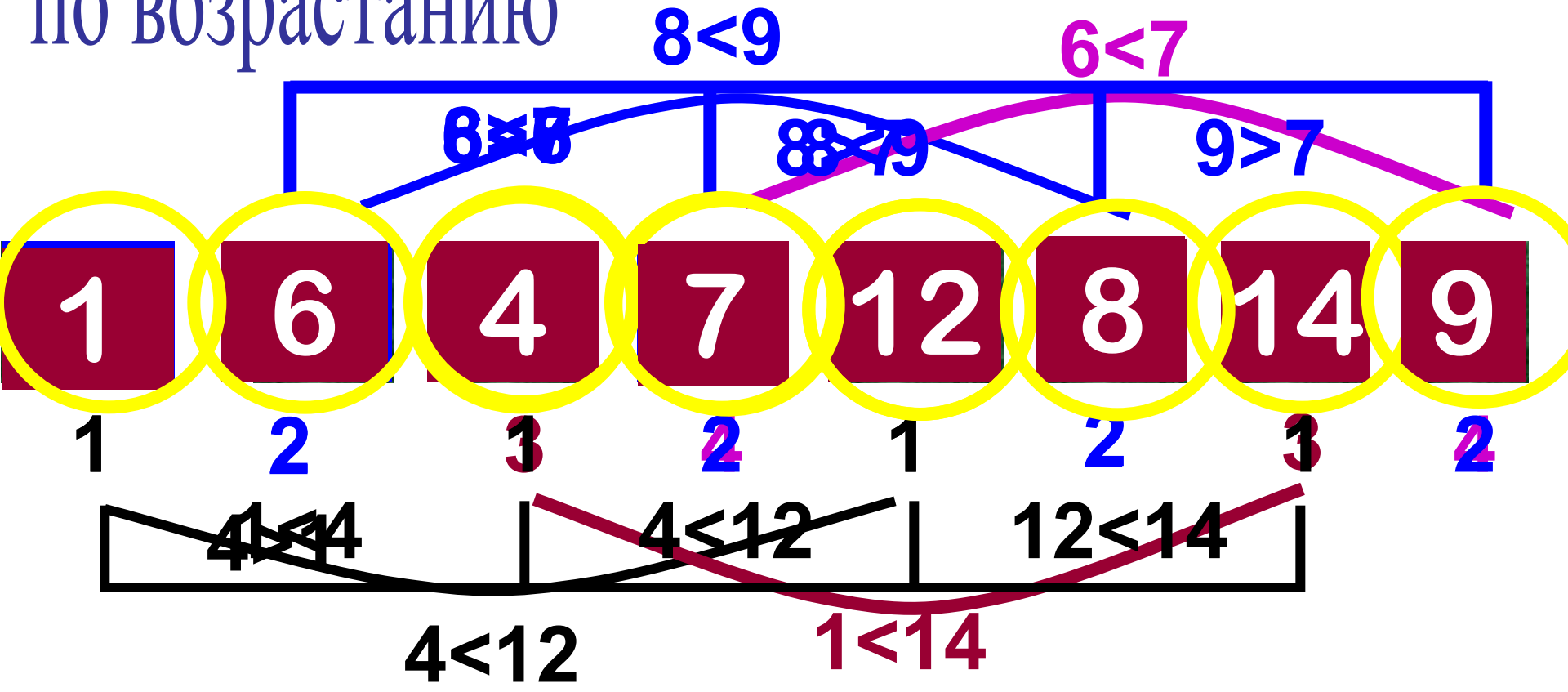
? Следует избегать последовательность, которые являются степенями 2 (т.е. нельзя использовать последовательность шагов – 4,2)

Суть сортировки:

- ✔ Сначала сортируются все элементы, отстоящие друг от друга на три позиции
- ✔ Затем сортируются элементы, расположенные на расстоянии двух позиций
- ✔ Наконец, сортируются все соседние элементы

Сортировка Шелла

2 шаг. 4 группы из 2-х элементов по возрастанию



Сортировка Шелла

3 шаг. 1 группа из 8-ми элементов

по возрастанию

Массив отсортирован по возрастанию

Постановка задачи

Пусть нужно отсортировать массив **A** по возрастанию, в котором **N** элементов методом Шелла

Вспомогательные переменные

j – номер первого элемента остатка.

i – номер перемещаемого элемента.

M- оптимальный шаг

P– промежуточное значение, используемое для перемещения элементов массива

Начало алгоритма.

Шаг 1. M=целая часть N/2

Шаг 2. Пока M<>0 **выполнять**

Шаг 2.1. i:=M+1

Шаг 2.2. Пока i<=N **выполнять**

Шаг 2.2.1. P=A[i]

Шаг 2.2.2. j=i-M

Шаг 2.2.3. Пока j>0 и P<A[j] **выполнять**

Шаг 2.2.3.1 A[j+M]=A[j]

Шаг 2.2.3.2 j=j-M

Шаг 2.2.4. A[j+M]=P

Шаг 2.2.5. i=i+1

Шаг 2.3. M=целая часть M/2

Конец алгоритма.

Алгоритм быстрой сортировки

- ✔ Придумана Ч.А.Р. Хоаром (Charles Antony Richard Hoare);
- ✔ В основе – сортировка обменами;
- ✔ Основана на делении массива

Суть сортировки:

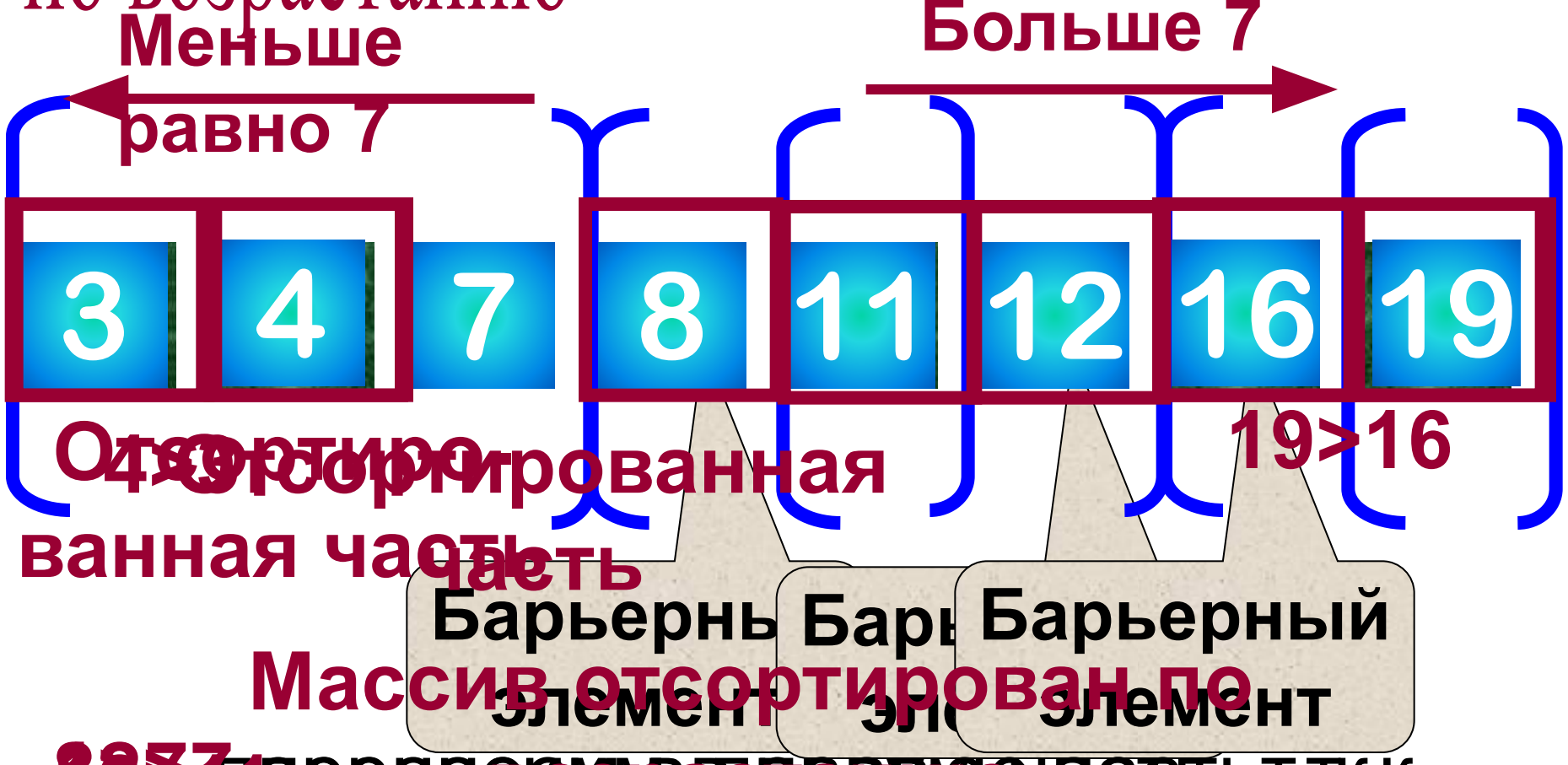
- Выбирается некоторое значение (x)- барьерный элемент, который определяется округлением до целого деления количества сортируемых элементов на 2;
- Просматриваем массив, двигаясь слева направо, пока не найдется элемент, больший x
- Затем просматриваем его справа налево, пока не найдется элемент, меньший x

Суть сортировки:

- Меняем найденные элементы местами. В случае, если не найден наибольший или наименьший элементы, местами меняется средний элемент с найденным наибольшим или наименьшим элементом;
- Дойдя до середины имеем 2 части массива;
- Процесс продолжается для каждой части, пока массив не будет отсортирован

Быстрая сортировка

по возрастанию



Массив отсортирован по возрастанию

переносим в правую часть, т. к.

не переносим,

поэтому меняем местами

Постановка задачи

Пусть нужно отсортировать массив **A** по возрастанию, в котором **n** элементов быстрым методом

Вспомогательные переменные:

t –конечный элемент массива

m - начальный элемент массива

x – элемент относительно которого перемещаются все остальные элементы.

w – промежуточное значение, используемое для перемещения элементов массива

Начало алгоритма.

Шаг 1 i=m j=t

Шаг 2 x=A[округление до целого(m+t)/2]

Шаг 3 **Пока** i<=j **выполнять**:

Шаг 3.1 **Если** A[i]<x **то** i:=i+1,

иначе

Если A[j]>x **то** j:=j-1

иначе

w:=A[i]; A[i]:=A[j]; A[j]:=w

i:=i+1, j:=j-1

Шаг 4 Если m<j то ***Алгоритм*** (A, m, j);

Шаг 5 Если i<t то ***Алгоритм*** (A, i, t).

Конец алгоритма.

Рекурсивный вызов процедуры