# Семинар №10
## ФАКИ 2016

Бирюков В. А.

November 17, 2016

# Алгоритмы

# Алгоритм

- Алгоритм – это формально описанная вычислительная процедура, получающая исходные данные, и выдающая результат вычислений на выход
(Кормен и др. "Алгоритмы: построение и анализ")

# Задача сортировки

- Задана последовательность чисел
- Нужно найти такую перестановку исходной последовательности, чтобы элементы были расположены по возрастанию
- 5 2 4 6 1 3 2 9 $\rightarrow$ 1 2 2 3 4 5 6 9

# Простейшие сортировки

- Сортировка вставками
- Сортировка выбором
- Сортировка пузырьком

# Анализ алгоритмов

# Анализ алгоритмов

- Обычно изучают зависимость времени работы от размера входа
- Размер входа – зависит от конкретной задачи
- Для сортировки, размер входа – это количество элементов, которые нужно отсортировать
- Время работы – число элементарных шагов, которые выполняет алгоритм

# Пример анализа

## Сортировка пузырьком

```
for (int j = 0; j < length-1; j++)
    for (int i = 0; i < length-1; i++)
        if (a[i] > a[i+1])
            swap(a[i], a[i+1]);
```

- Число операций, требуемых на один проход: $a * n$
- Число проходов: $n$
- Значит, время работы $\sim n^2$

# Принцип "разделяй и влавствуй"

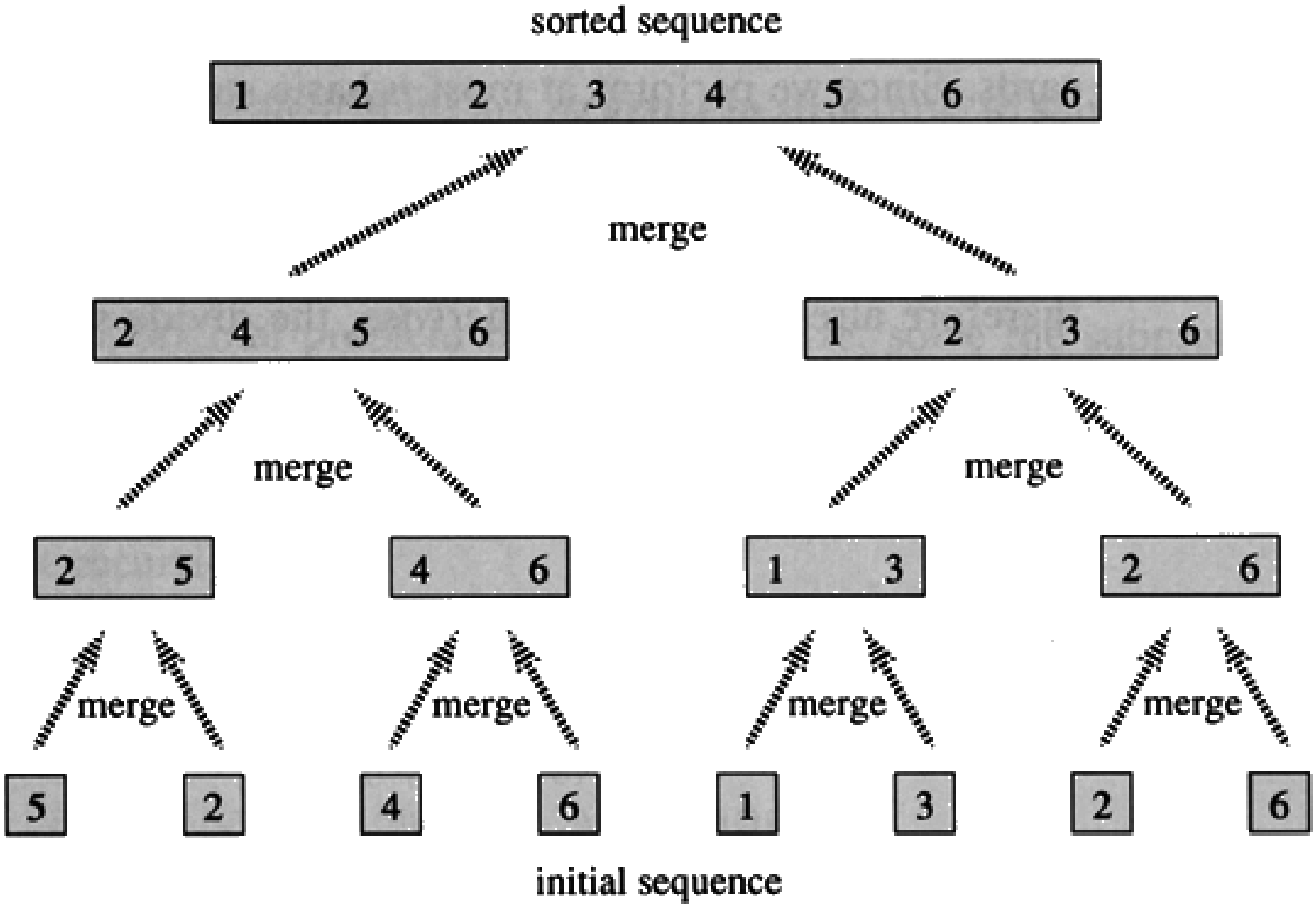
# Принцип "разделяй и влавствуй"

- Задача разбивается на несколько подзадач меньшего размера
- Эти задачи решаются (обычно с помощью рекурсивного вызова)
- Решения этих задач комбинируются и получается решение исходной задачи

# Сортировка слиянием

- Разбиваем массив на 2 половины
- Сортируем каждую половину
- Соединяем 2 упорядоченных массива в один

# Сортировка слиянием



# Сортировка слиянием

```
void MergeSort(int * A, int p, int r)
{
    if (p < r)
    {
        int q = (p + r) / 2;
        MergeSort(A, p, q);
        MergeSort(A, q + 1, r);
        Merge(A, p, q, r);
    }
}
```

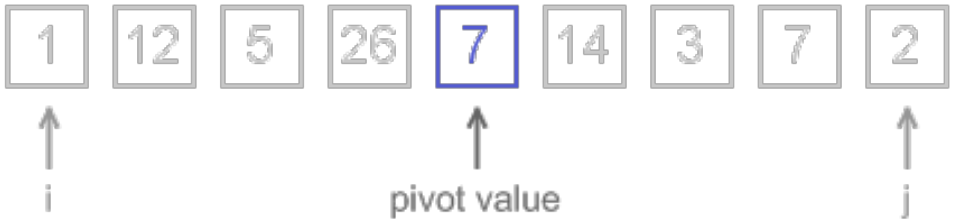
# Быстрая сортировка

# Быстрая сортировка (quicksort)

- Выбираем в массиве некоторый элемент, который будем называть опорным
- Переставляем элементы массива таким образом, чтобы все элементы со значением меньшим или равным опорному элементу, оказались слева от него, а все элементы, превышающие по значению опорный — справа от него
- Рекурсивно сортируем подмассивы, лежащие слева и справа от опорного элемента

# Быстрая сортировка (quicksort)


Неотсортированный массив

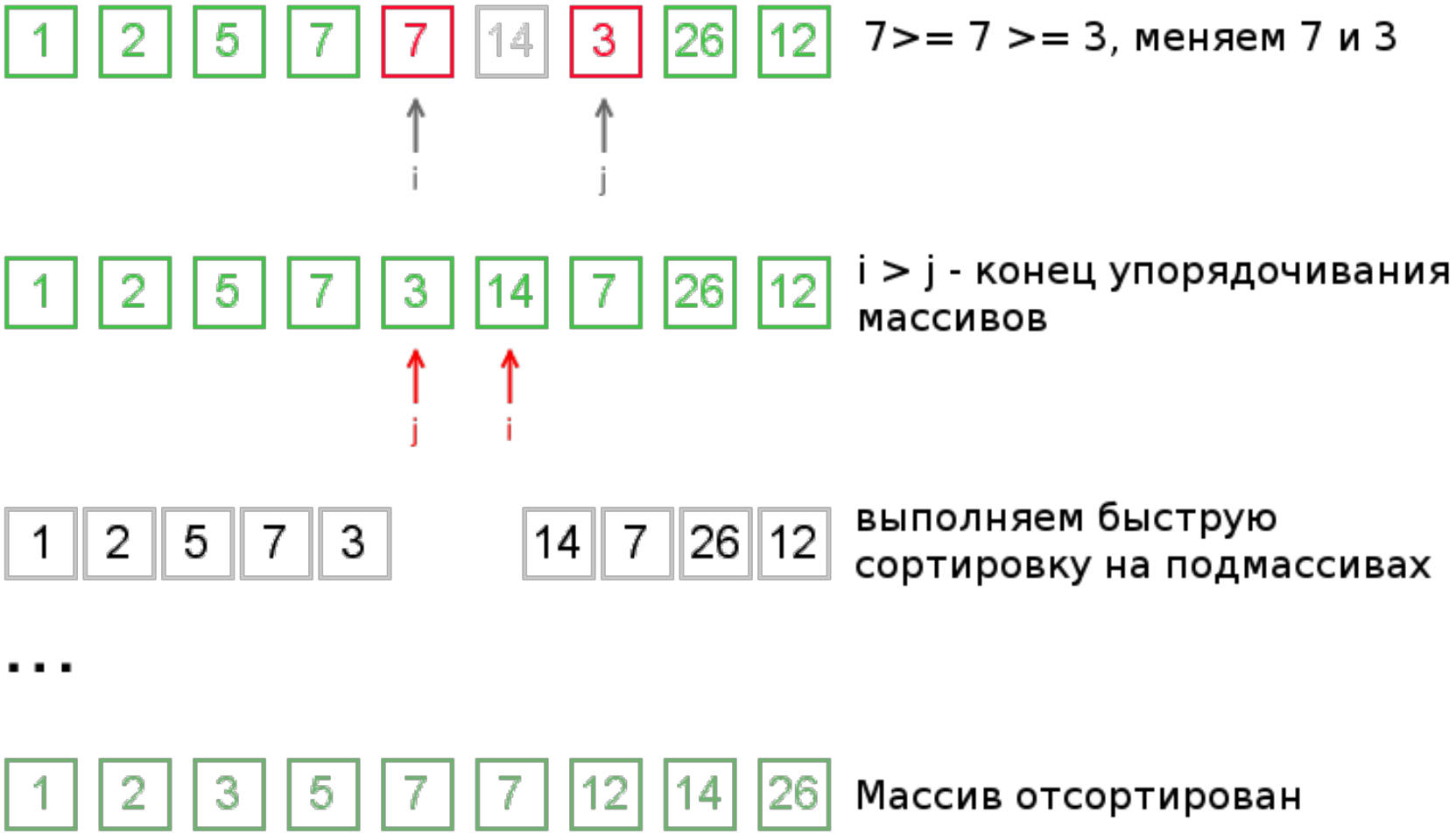



Опорное значение = 7


12 >= 7 >= 2,
меняем местами 12 и 2


26 >= 7 >= 7
меняем 26 и 7

# Быстрая сортировка (quicksort)



# Время работы сортировок

# Время работы сортировок

- Время работы сортировки пузырьком, выбором и вставками $\sim n^2$
- Время работы сортировки слиянием и быстрой сортировки в среднем $\sim n\log(n)$

# Время работы сортировок

- Пусть мы хотим отсортировать массив из 1 млн. чисел
- Сортировка пузырьком написана аккуратно и требует $2n^2$ операций и выполняется на суперкомпьютере(x100)
- Сортировка слиянием написана неэффективно и требует 50nlog(n) операций и выполняется на пк(x1)
- Сортировка пузырьком выполнится за 5.5 часов
- Сортировка слиянием выполнится за 17 минут

# Стандартная сортировка qsort()

# Стандартная сортировка qsort()

```
#include <stdlib.h>
int values[] = { 88, 56, 100, 2, 25 };

int cmp(const void * a, const void * b)
{
   return ( *(int*)a - *(int*)b );
}
...
qsort(values, 5, sizeof(int), cmp);
```

# Задание

# Задание

- bubble sort
- quick sort
- Задачи на qsort: Станция Новодачная-сортировочная