ФГОБУ ВПО "СибГУТИ"
Кафедра вычислительных систем

ЯЗЫКИ ПРОГРАММИРОВАНИЯ / ПРОГРАММИРОВАНИЕ

Алгоритмы внешней сортировки (часть I, базовые понятия и алгоритмы)

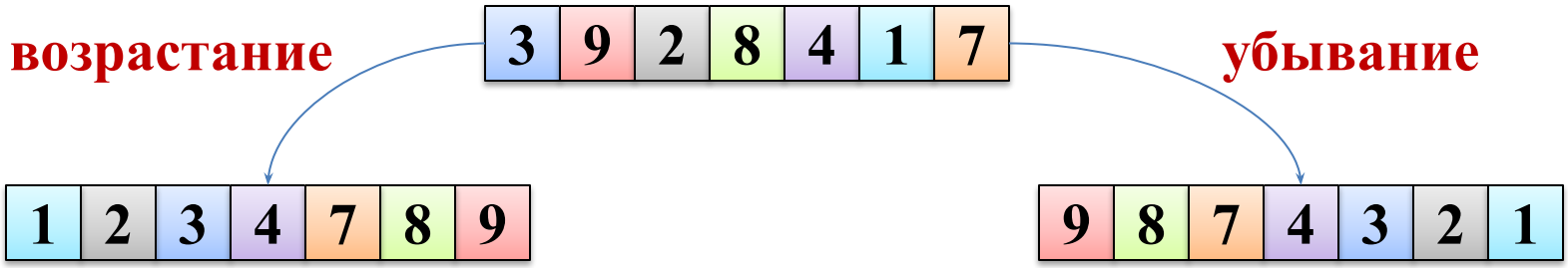
Преподаватель:

Доцент Кафедры ВС, к.т.н.

Поляков Артем Юрьевич

Сортировка данных

- ***Сортировка*** – процесс перестановки элементов некоторого множества в определенном порядке.



- ***Цель сортировки*** – упрощение поиска данных в этом множестве.
- Задача сортировки данных часто возникает при разработке программного обеспечения.
- Алгоритмы сортировки можно разделить на:
  - алгоритмы **внутренней** сортировки;
  - алгоритмы **внешней** сортировки.

Оценка алгоритмов сортировки (1)

Время (вычислительная сложность) – основной параметр, характеризующий быстродействие алгоритма.

При анализе алгоритмов обычно учитывают *худшее*, *среднее* и *лучшее* поведение алгоритма на наборе допустимых входных данных размера n (n – мощность входного множества A: $n = |A|$).

Для типичного алгоритма сортировки *хорошее* поведение – это $O(n \cdot \log n)$ и *плохое* поведение — это $O(n^2)$.

При оценке *алгоритмов сортировки* учитывается:

- C – количество операций сравнения;
- M – количество операций пересылки данных.

Оценка алгоритмов сортировки (2)

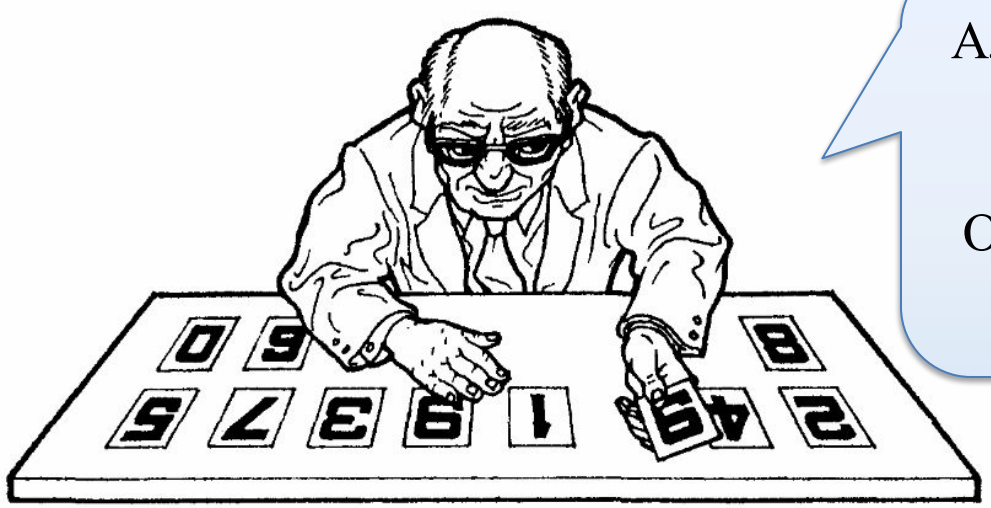
Память – ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных.

При оценке *не учитывается*:

- место, которое занимает исходный массив
- независящие от входной последовательности затраты, например, на хранение кода программы. Считается, что такие расходы требуют $O(1)$ памяти.

Алгоритмы сортировки, не потребляющие дополнительной памяти, относят к сортировкам на месте.

Алгоритмы внешней и внутренней сортировки



Алгоритмы **внутренней** сортировки оперируют сравнительно небольшими объемами данных. Они могут "видеть" любой элемент сортируемого множества

Алгоритмы внешней сортировки применяются тогда, когда количество элементов велико и нет возможности "разложить их на столе" (в оперативной памяти).

Алгоритмы внутренней сортировки

Алгоритмы внутренней сортировки (сортировки массивов) предназначены для работы с данными, которые полностью помещаются в **оперативную память** вычислительной машины, выполняющей данную операцию.

Для оперативной памяти характерно приблизительно одинаковое время доступа ко всем ее элементам.

Поэтому важной характеристикой алгоритмов внутренней сортировки является экономичность по памяти.

Алгоритмы внешней сортировки

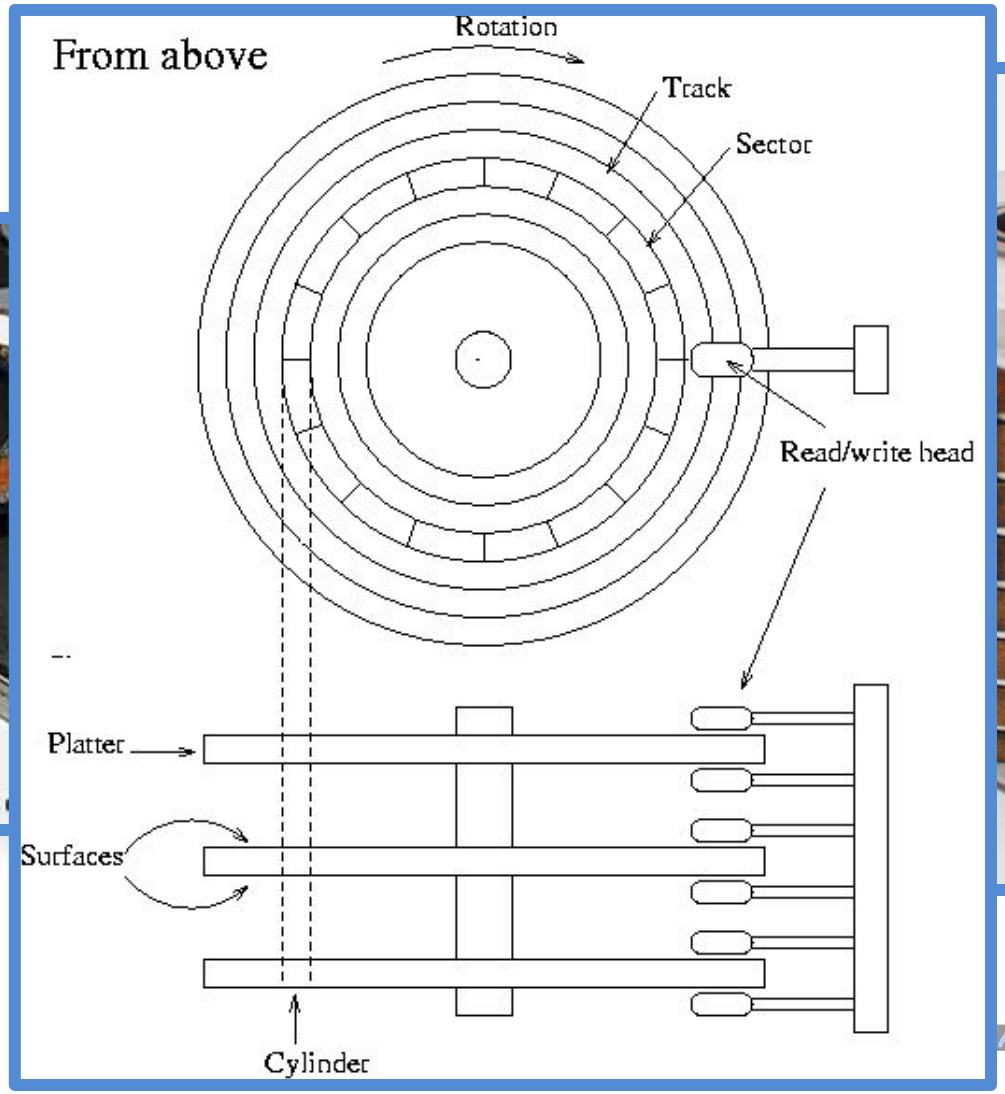
Алгоритмы внешней сортировки (сортировки файлов) предназначены для работы с данными, объем которых не позволяет полностью разместить их в **оперативной памяти** вычислительной машины, выполняющей данную операцию.

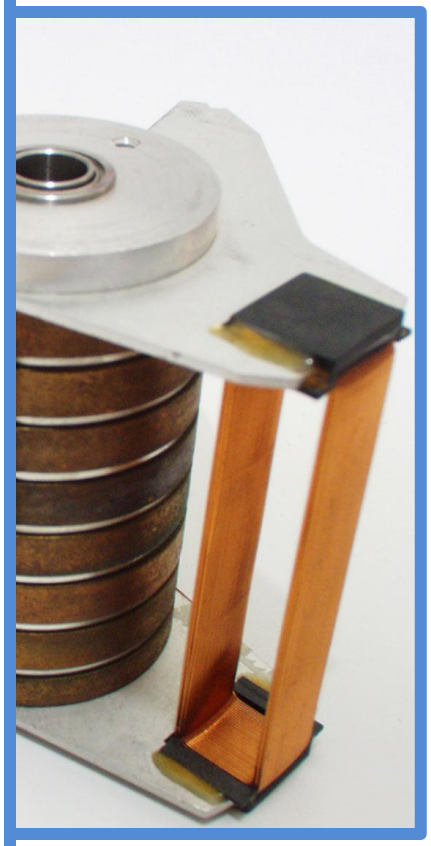
Входные данные располагаются на внешних запоминающих устройствах, таких как диски и магнитные ленты.

Внешняя память характеризуется **последовательным доступом** к ее элементам. **В каждый момент времени имеется непосредственный доступ только к одному элементу**.

Основной метод сортировки – слияние.

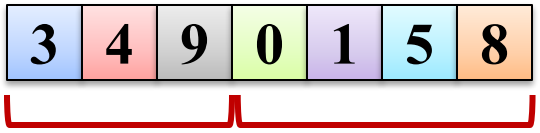
Накопители на жестких магнитных дисках



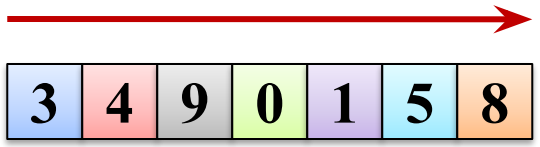


Терминология

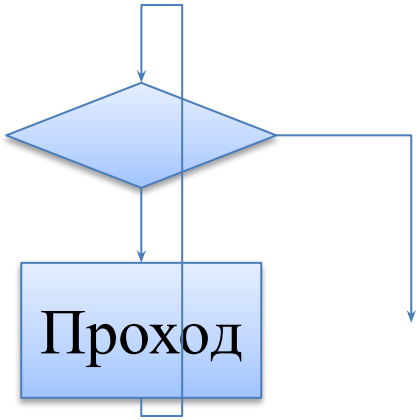
Серия – упорядоченная подпоследовательность максимальной длины.



Фаза – операция однократной обработки всего набора данных

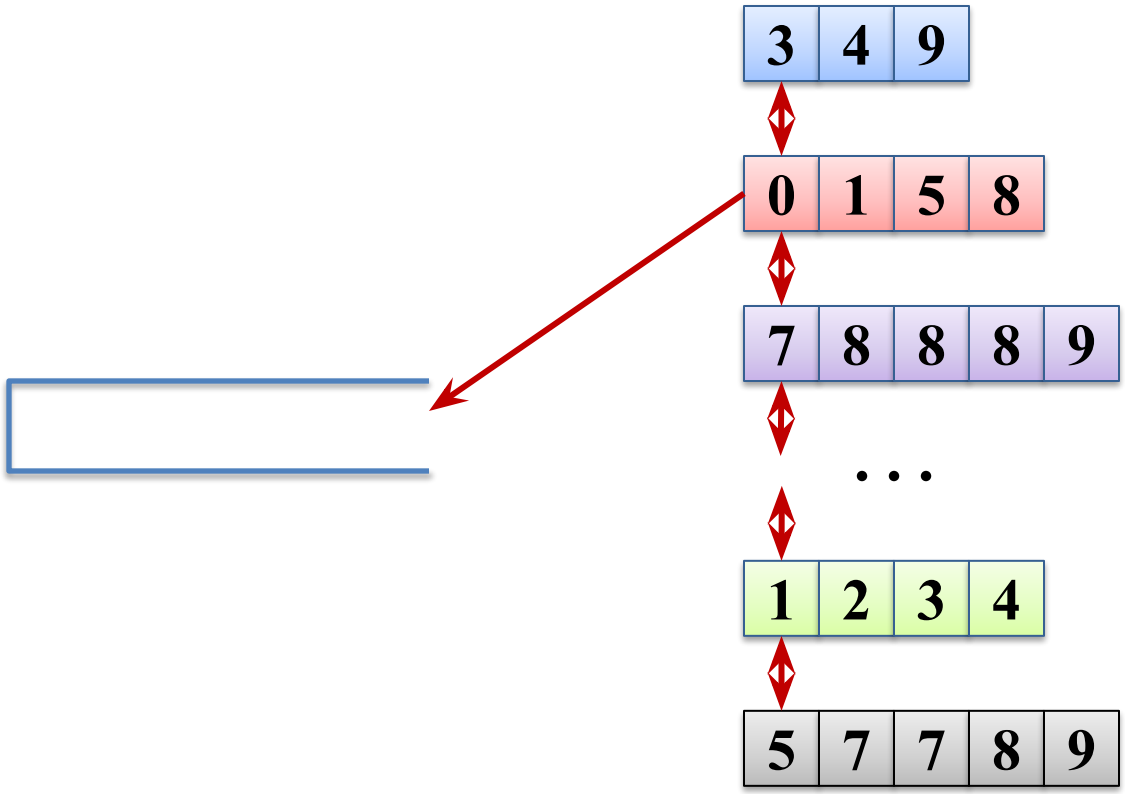


Проход (***этап***) – наименьший процесс, повторение которого образует процесс сортировки.



Процедура слияния серий

Процедура слияния (2)



Задание

Выполнить слияние следующих последовательностей:

4 1 8 4 12 43 19 21

9 10 11 4 3 22 17

7 12 3 33 21 15 32 8

10 9 8 12 13 14 15

Слияние выполнять отдельно по сериям

Сколько серий в полученной последовательности?

Задание

Выполнить слияние следующих последовательностей:

4 1 8 4 12 43 19 21

9 10 11 4 3 22 17

7 12 3 33 21 15 32 8

10 9 8 12 13 14 15

4 7 9 10 10 11 12 ' 1 3 4 8 9 33 '

3 4 12 12 13 14 15 21 22 43 ' 15 17 19 21 32 ' 8

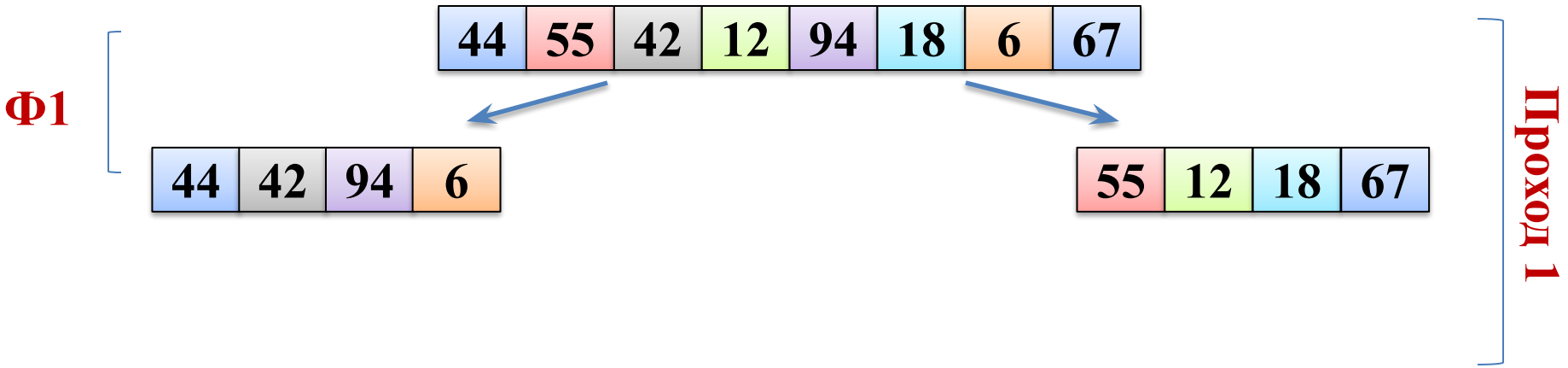
ПРОСТАЯ СОРТИРОВКА СЛИЯНИЕМ

(STRAIGHT MERGE)

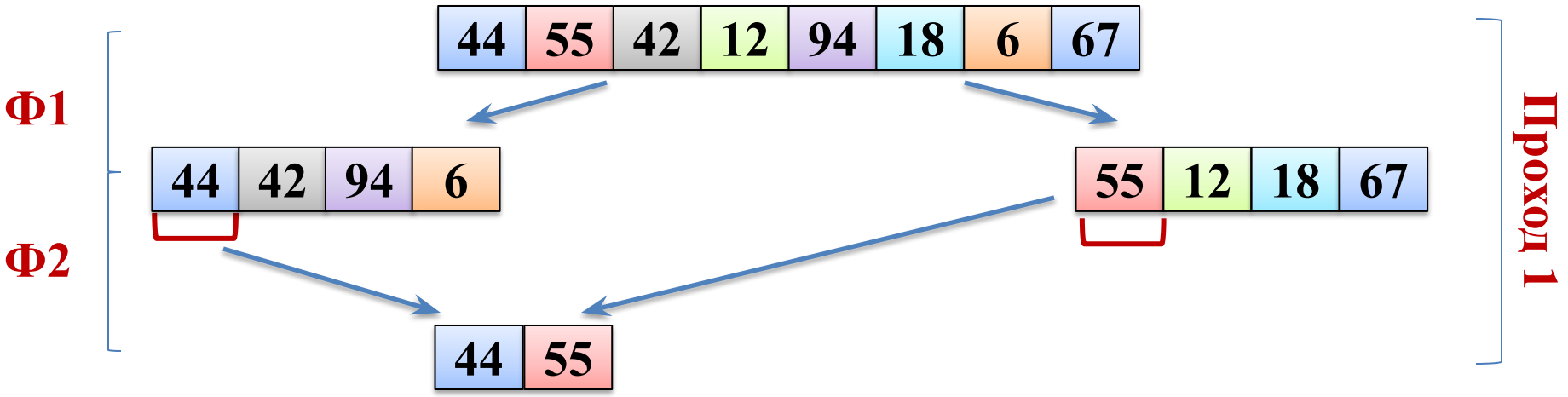
Простая сортировка слиянием (двухфазная) (straight merge)

44	55	42	12	94	18	6	67

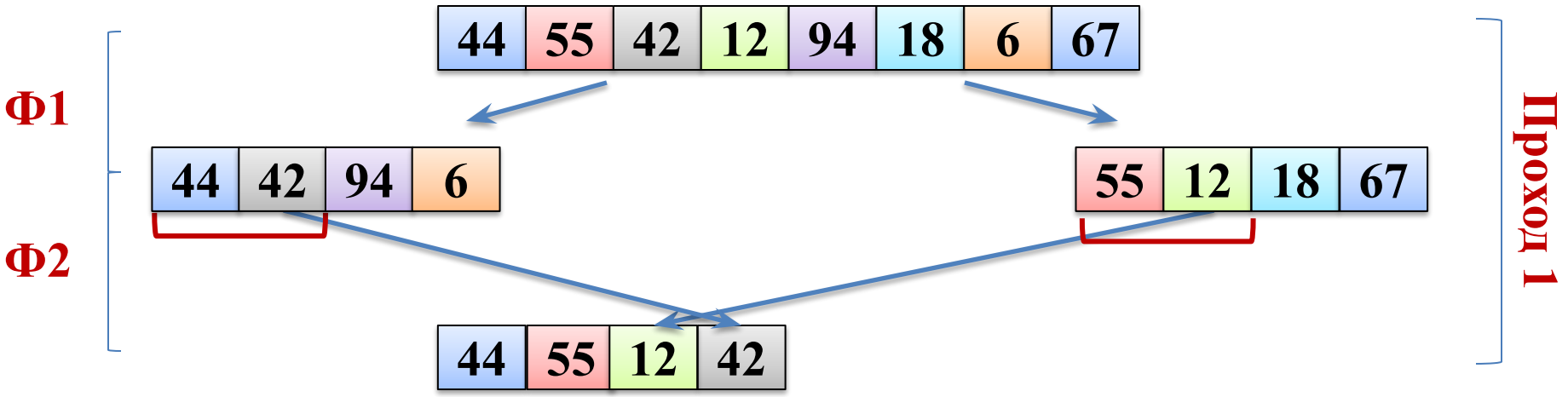
Простая сортировка слиянием (двухфазная) (straight merge)



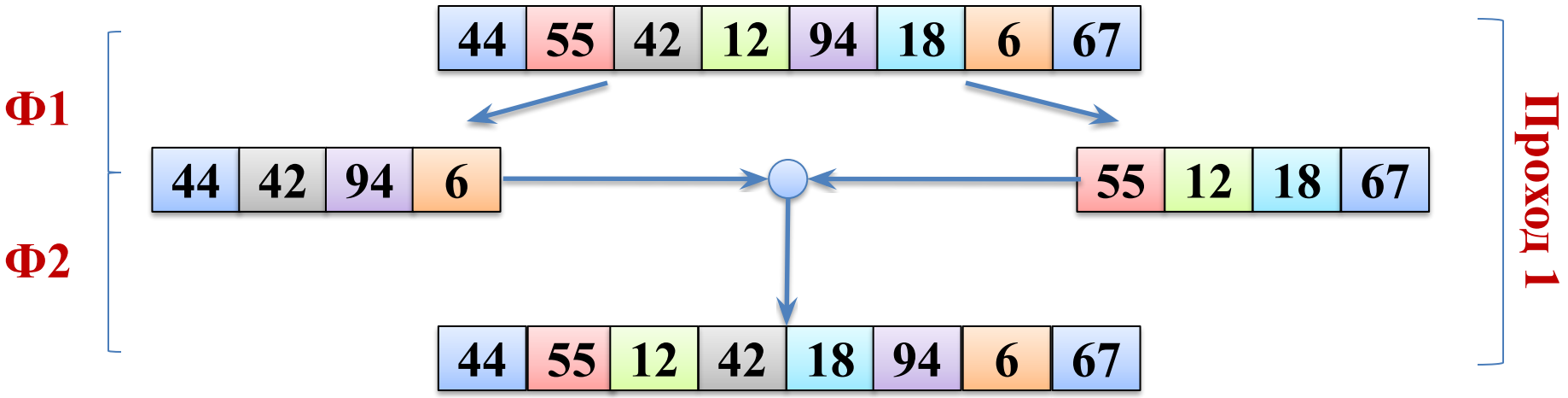
Простая сортировка слиянием (двухфазная) (straight merge)



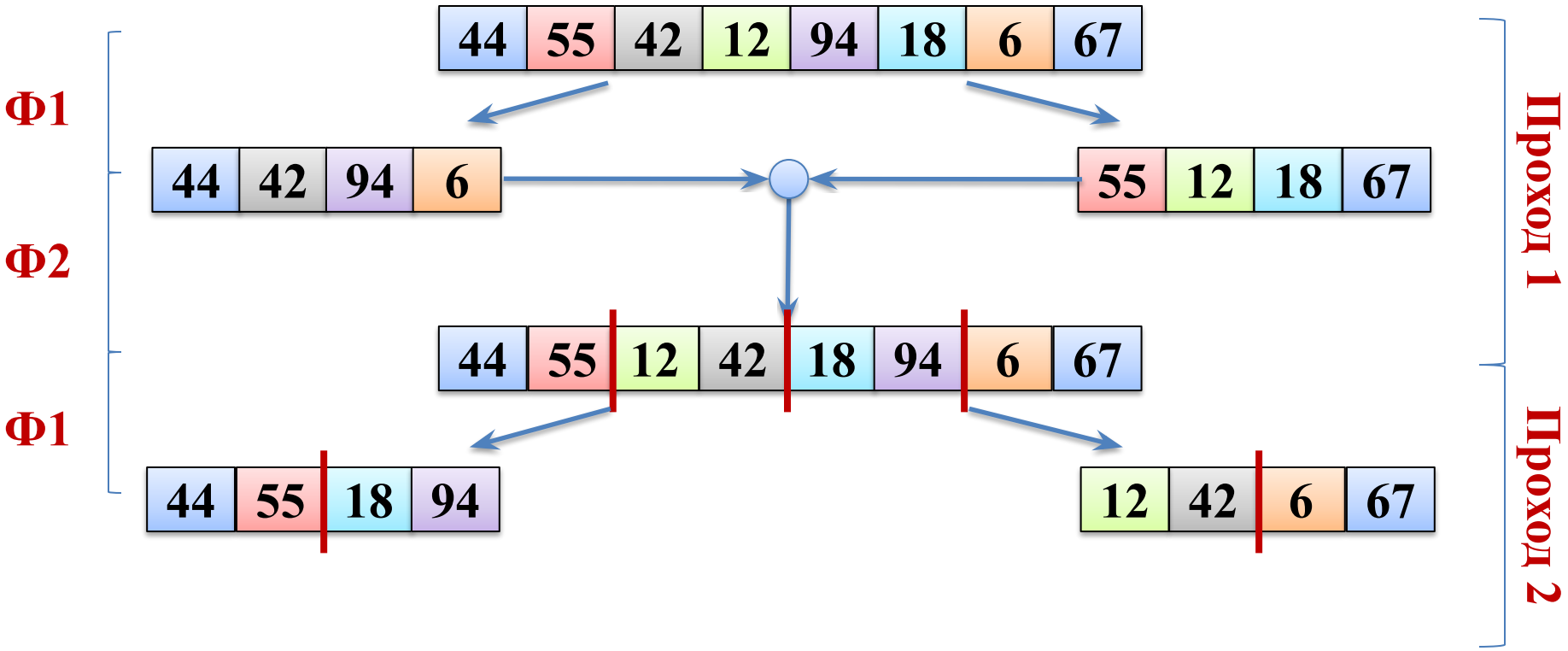
Простая сортировка слиянием (двухфазная) (straight merge)



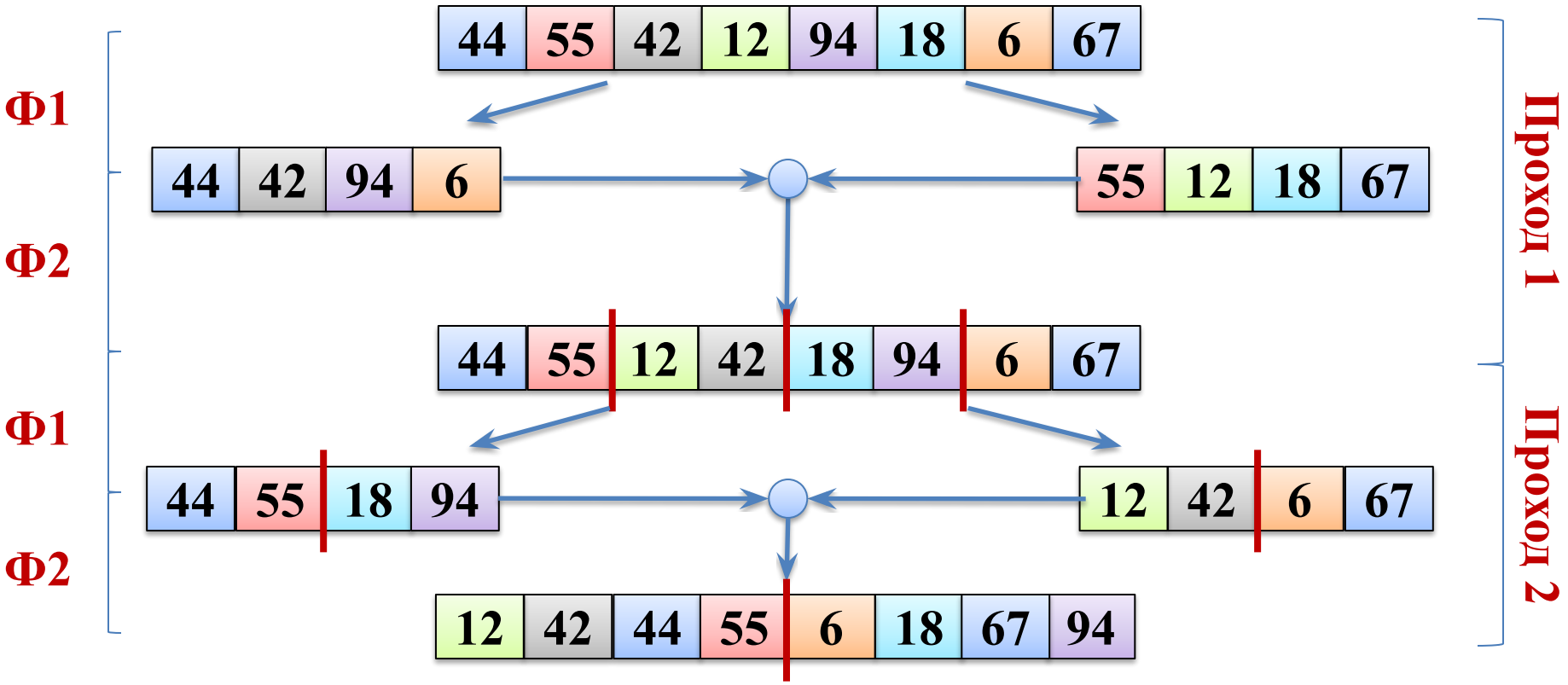
Простая сортировка слиянием (двухфазная) (straight merge)



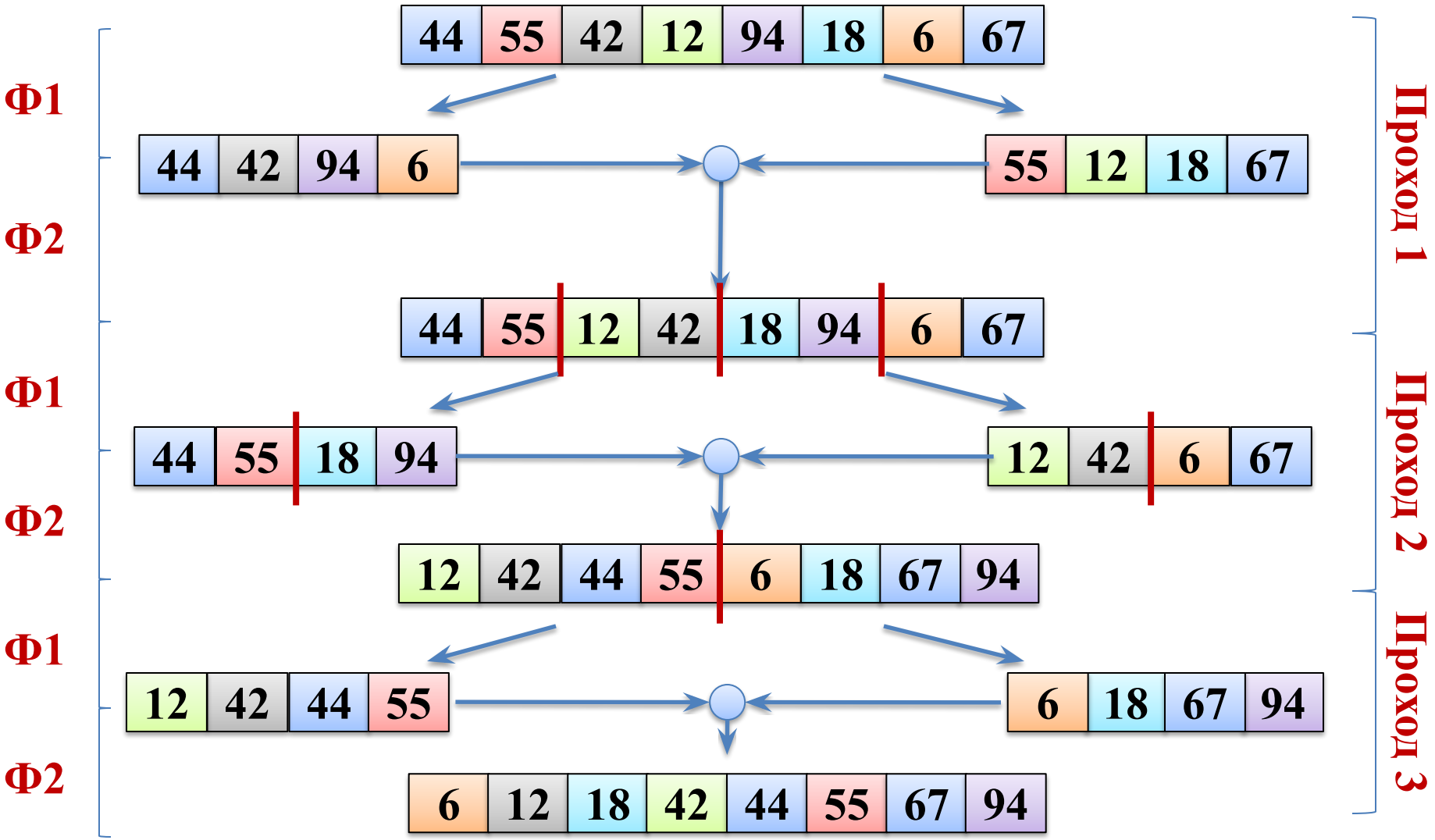
Простая сортировка слиянием (двухфазная) (straight merge)



Простая сортировка слиянием (двухфазная) (straight merge)



Простая сортировка слиянием (двухфазная) (straight merge)



Алгоритм простой сортировки слиянием

```
ввод n, a
k ← 1
while k < n do
    (b, c) ← DISTR(a, n, k)
    a ← MERGE(b, c,  k)
    k ← k · 2
done
```

```
DISTRIBUTE(a, n, k)
    i ← 1, b ← < >, c ← < >
    while i < n do
        k ← min(n – i, k)
        b ← b & <a_i, …, a_{i+k} >
        b ↔ c
        i ← i + k
    return (b, c)
```

```
MERGE(b, c, k)
    while b ≠ < > ИЛИ c ≠ < > do
        f1 ← first(b), b ← rest(b), n1 ← 1
        f2 ← first(c), c ← rest(c), n2 ← 1
        while b ≠ < > И c ≠ < > И
              n1 ≤ k И n2 ≤ k  do
            if f1 < f2 then
                a ← a & f1, n1 ← n1 + 1
                f1 ← first(b), b ← rest(b)
            else
                a ← a & f2, n2 ← n2 + 1
                f2 ← first(c), c ← rest(c)
        while b ≠ < > И n1 ≤ k do n1 ← n1 + 1,
            a ← a & first (b), b ← rest(b)
        while c ≠ < > И n2 ≤ k do n2 ← n2 + 1,
            a ← a & first (c) , c ← rest(c)
```

Реализация алгоритма распределения

DISTRIBUTE(a, n, k)
  $i \leftarrow 1$, $b \leftarrow <>$, $c \leftarrow <>$
  while $i < n$ **do**
    $k \leftarrow$ **min**$(n - i, k)$
    $b \leftarrow b$ & $<a_i, \ldots, a_{i+k}>$
    $b \leftrightarrow c$
    $i \leftarrow i + k$
  return (b, c)

```
void distr(int s[], int n, int d1[], int *d1n,
           int d2[], int *d2n, int k)
{
    int *wptr = d1, *bptr = d2, *tp;
    int *wn = d1n, *bn = d2n, i = 0, j, *tn;
    *wn = 0; *bn = 0;
    while( i < n ){
        k = (k < (n - i)) ? k : n - i; // min(k,n-i)
        for(j=0; j<k; j++){
            wptr[(*wn)++] = s[i++];
        }
        tp = wptr; wptr = bptr; bptr = tp;
        tn = wn; wn = bn; bn = tn;
    }
}
```

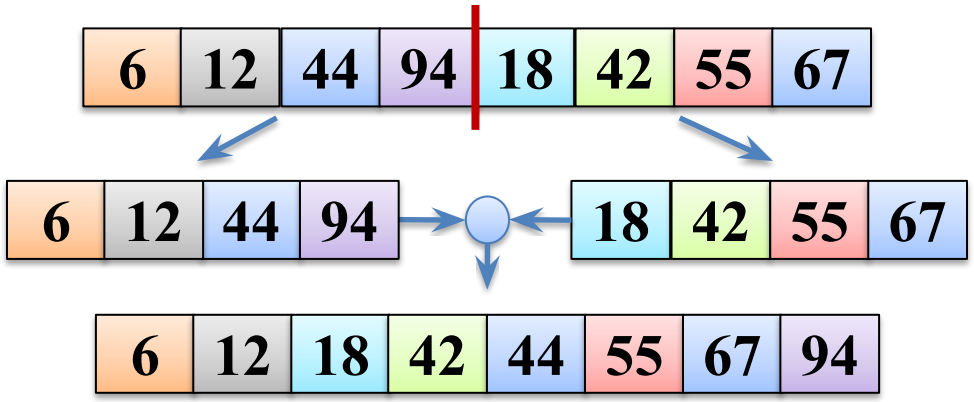
Алгоритм простой сортировки слиянием

ввод n, a
$k \leftarrow 1$
while $k < n$ **do**
    $(b, c) \leftarrow \text{DISTR}(a)$
    $a \leftarrow \text{MERGE}(b, c)$
    $k \leftarrow k \cdot 2$
done

Выполнить сортировку последовательности:
91 4 18 26 44 21 30 19 81 16 45 200 57 71 82 99 150 212

Анализ алгоритма простой сортировки слиянием

```
ввод n, a
k ← 1
while k < n do
    (b, c) ← DISTR(a)
    a ← MERGE(b, c)
    k ← k · 2
done
```



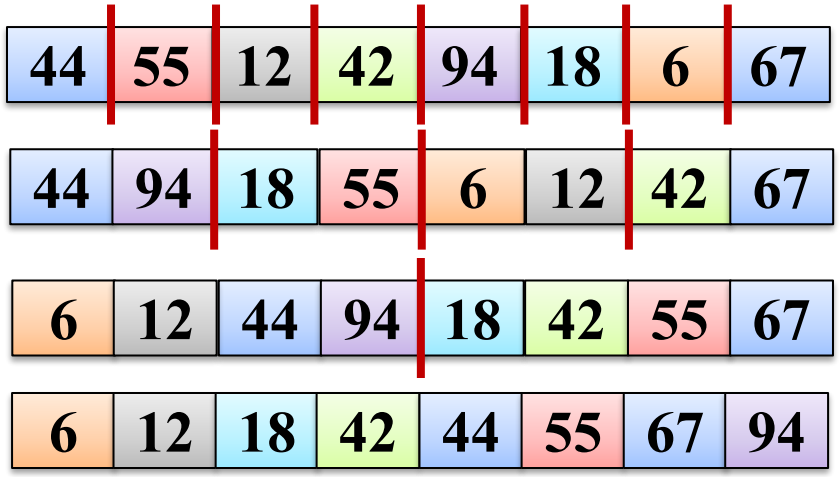
Количество *R* пересылок данных на одном этапе:

- этап состоит из двух фаз, на каждой из которых выполняется копирование всех элементов из *a*;
- Количество *R* элементов, обраб. на одном этапе:

$$R = 2n$$

Анализ алгоритма простой сортировки слиянием

```
ввод n, a
k ← 1
while k < n do
    (b, c) ← DISTR(a)
    a ← MERGE(b, c)
    k ← k · 2
done
```



Количество *i* этапов:

- зависит от параметра k (***длина серии***), который на каждом этапе удваивается;
- общее число i этапов:

$$i = [\log_2 n] + 1$$

1) $k = 1$ (2^0)
2) $k = 2$ (2^1)
3) $k = 4$ (2^2)

i) $k = 2^{i-1} < n$
i+1) $k = 2^i \geq n$

Анализ алгоритма простой сортировки слиянием

```
ввод n, a
k ← 1
while k < n do
    (b, c) ← DISTR(a)
    a ← MERGE(b, c)
    k ← k · 2
done
```

Количество *M* пересылок:

$$M = i \cdot R = 2n \cdot [\log_2 n] = O(n \cdot \log_2 n)$$

Число *C* сравнений по ключу:

- сопоставимо с *M*;
- время сравнения значительно ниже времени пересылки.

Алгоритм требует $n = O(n)$ дополнительной памяти

Недостатки алгоритма простой сортировки слиянием

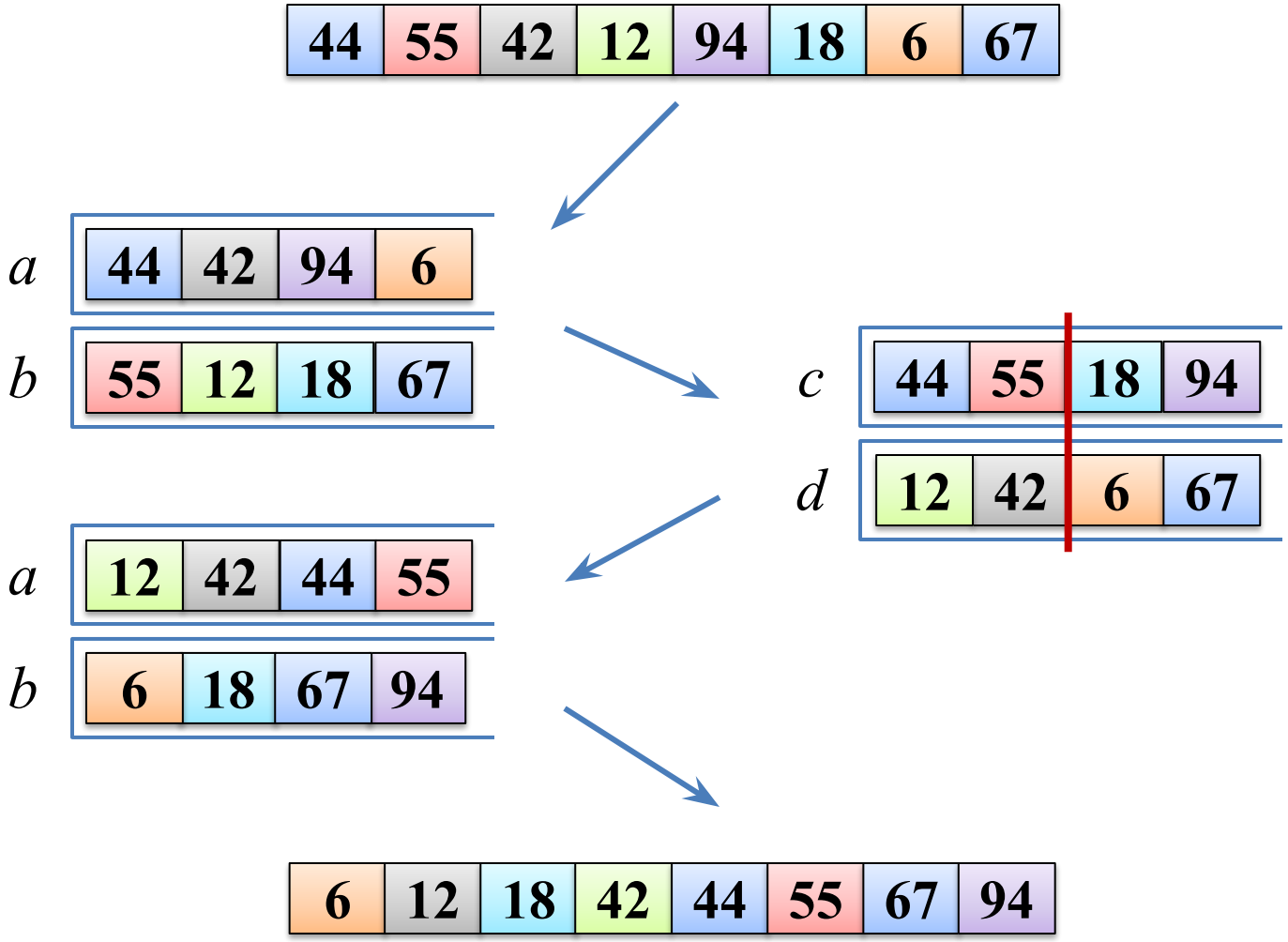
Доступ к данным на внешнем устройстве занимает существенное время

Фаза разбиения последовательности по вспомогательным файлам не вносит вклада в сортировку (переупорядочивания элементов не происходит)

МЕТОД СБАЛАНСИРОВАННЫХ СЛИЯНИЙ

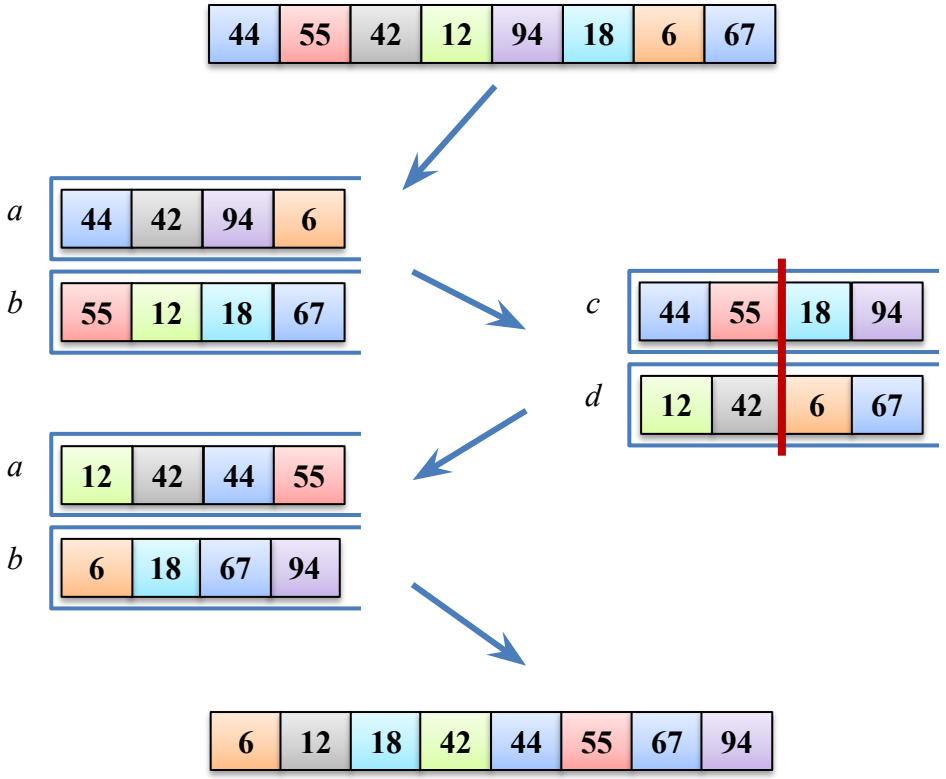
(BALANCED MERGE)

Метод сбалансированных слияний (balanced merge)



Алгоритм сбалансированных слияний

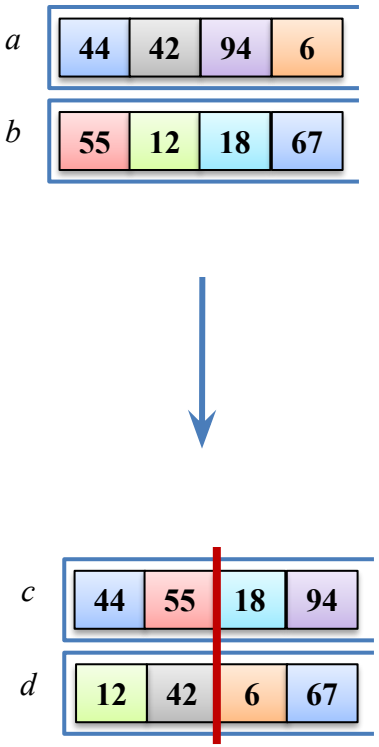
```
ввод n, a
k ← 1
(a, b) ← DISTR (a, n, k)
while k < n do
    (c, d) ← MERGE2(a, b, k)
    (a, b) ↔ (c, d)
    k ← k · 2
done
(a, b) ↔ (c, d)
a← MERGE(b, c, k)
```



Предложите алгоритм процедуры MERGE2

Объединенная процедура слияния и распределения

MERGE2(s_1, s_2, k)
  $d_1 \leftarrow <>$, $d_2 \leftarrow <>$, $f_1 \leftarrow$ **first**(s_1), $f_2 \leftarrow$ **first**(s_2),
  $s_1 \leftarrow$ **rest**(s_1), $s_2 \leftarrow$ **rest**(s_2)
  while $s_1 \neq <>$ ИЛИ $s_2 \neq <>$ **do**
    $n_1 \leftarrow 1$, $n_2 \leftarrow 1$
    while ($s_1 \neq <>$ И $n_1 \leq k$) И ($s_2 \neq <>$ И $n_2 \leq k$) **do**
      if $f_1 < f_2$ **then** $d_1 \leftarrow d_1 \,\&\, f_1$, $n_1 \leftarrow n_1 + 1$,
        $f_1 \leftarrow$ **first**(s_1), $s_1 \leftarrow$ **rest**(s_1)
      else $d_1 \leftarrow d_1 \,\&\, f_2$, $n_2 \leftarrow n_2 + 1$,
        $f_2 \leftarrow$ **first**(s_2), $s_2 \leftarrow$ **rest**(s_2)
    while $s_1 \neq <>$ И $n_1 \leq k$ **do**
      $d_1 \leftarrow d_1$ & **first** (s_1)
    while $s_2 \neq <>$ И $n_2 \leq k$ **do**
      $d_1 \leftarrow d_1$ & **first** (s_2)
    $d_2 \leftrightarrow d_1$



Анализ алгоритма сбалансированной сортировки слиянием

```
ввод n, a
k ← 1
(a, b) ← DISTR(a, n, k)
while k < n do
    (c, d) ← MERGE2(a, b, k)
    (a, b) ↔ (c, d)
    k ← k · 2
done
(a, b) ↔ (c, d)
a← MERGE(b, c, k)
```

Оцените количество пересылок данных, которое производится при применении алгоритма сбалансированной сортировки слиянием

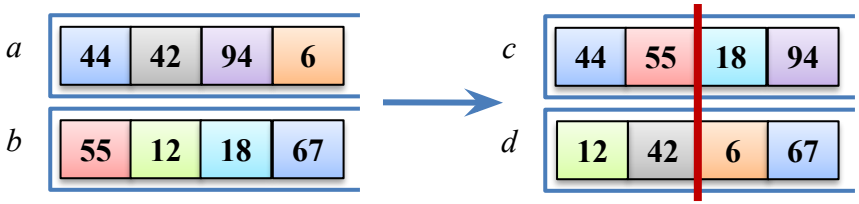
Анализ алгоритма сбалансированной сортировки слиянием

```
ввод n, a
k ← 1
(a, b) ← DISTR(a, n, k)
while k < n do
    (c, d) ← MERGE2(a, b, k)
    (a, b) ↔ (c, d)
    k ← k · 2
done
(a, b) ↔ (c, d)
a← MERGE(b, c, k)
```

Количество *R* пересылок данных на одном этапе:

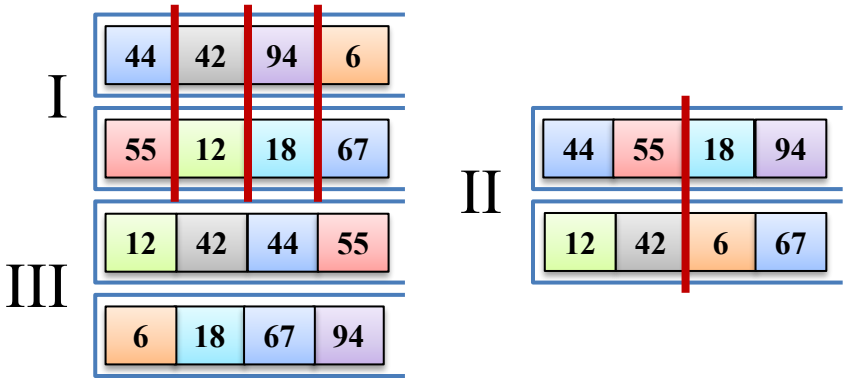
- этап состоит из одной фазы, в рамках которой выполняется копирование каждого элемента *a*;
- Количество *M* пересылок на одном этапе:

$$R = n$$



Анализ алгоритма сбалансированной сортировки слиянием

ввод n, a
$k \leftarrow 1$
$(a, b) \leftarrow \text{DISTR}(a, n, k)$
while $k < n$ **do**
    $(c, d) \leftarrow \text{MERGE2}(a, b, k)$
    $(a, b) \leftrightarrow (c, d)$
    $k \leftarrow k \cdot 2$
done
$(a, b) \leftrightarrow (c, d)$
$a \leftarrow \text{MERGE}(b, c, k)$



Количество *i* этапов:

- зависит от параметра k (***длина серии***), который на каждом этапе удваивается;
- общее число i этапов:

$$i = [\log_2 n] + 1$$

1) $k = 1$ (2^0)
2) $k = 2$ (2^1)
3) $k = 4$ (2^2)

i) $k = 2^{i-1} < n$
i+1) $k = 2^i \geq n$

Анализ алгоритма сбалансированной сортировки слиянием

```
ввод n, a
k ← 1
(a, b) ← DISTR(a, n, k)
while k < n do
    (c, d) ← MERGE2(a, b, k)
    (a, b) ↔ (c, d)
    k ← k · 2
done
(a, b) ↔ (c, d)
a← MERGE(b, c, k)
```

Количество *M* пересылок:

$$M = i \cdot R = n \cdot [\log_2 n] = O(n \cdot \log_2 n)$$

Число *C* сравнений по ключу:

- сопоставимо с *M*;
- время сравнения значительно ниже времени пересылки.

Алгоритм требует $2n = O(n)$ дополнительной памяти

Недостатки алгоритмов простой и сбалансированной сортировки слиянием

Доступ к данным на внешнем устройстве занимает существенное время

Рассмотренные алгоритмы сортировки не обладают свойством естественности поведения.

Естественность поведения – эффективность метода при обработке уже упорядоченных или частично упорядоченных данных. Алгоритм ведёт себя естественно, если учитывает эту характеристику входной последовательности и работает лучше.

Литература

1. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона / Пер. с англ. Ткачев Ф.В. – М.: ДМК Пресс, 2012 г. – 272 с.,
2. Кнут, Д.Э. Искусство программирования: в 3 т. Т. 3. Сортировка и поиск [Текст] : [учеб. пособие]; пер. с англ. / под общ. ред. Ю.В. Казаченко. - 3-е изд. – М.: Издат.дом "Вильямс", 2010. – 822с.
3. Седжвик Р. Алгоритмы на C++ (Algorithms in C++): Пер. с англ. – М.: Издательский дом "Вильямс", 2011 г. – 1056 с. – ISBN 978-5-8459-1650-1, 978-0-321-60633-4;