Метод прямого слияния MergeSort

Дан неупорядоченный список **S**.

В основе алгоритма лежит **операция слияния серий**.

Определение. p-серией называется неубывающая последовательность из **p** элементов.

Задача: Имеется две упорядоченные последовательности **a** и **b** размером **q** и **r** соответственно. Необходимо получить последовательность **c** путем слияния **a** и **b**, длина последовательности **c** будет равна **q + r**.

Пример:

a: 1 4 5 6’

b: 2 3 6” 7 8

c: 1 2 3 4 6” 7 8

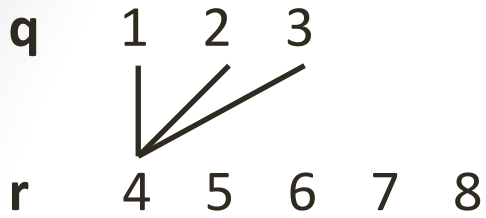
Алгоритм слияния серий

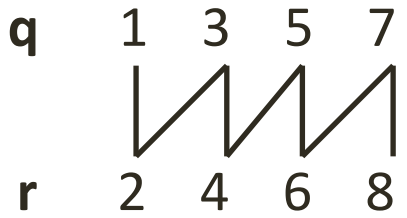
Слияние q–серии из списка a с r–серией из списка b, запись результата в очередь c

```
DO ( q ≠ 0  и  r ≠ 0 )
    IF ( a→Data ≤ b→Data)
        < Переместить элемент из списка a в очередь c >
        q := q-1
    ELSE
        < Переместить элемент из списка b в очередь c >
        r := r-1
    FI
OD
DO ( q > 0 )
    < Переместить элемент из списка a в очередь c >
    q := q-1
OD
DO ( r > 0 )
    < Переместить элемент из списка b в очередь c >
    r := r-1
OD
```

Трудоемкость алгоритма слияния серий

Трудоемкость зависит от расположения элементов в сериях.





Количество сравнений: $\min(q, r) \le C \le q+r-1$

Количество перестановок: $M = q+r$

Метод прямого слияния (MergeSort)

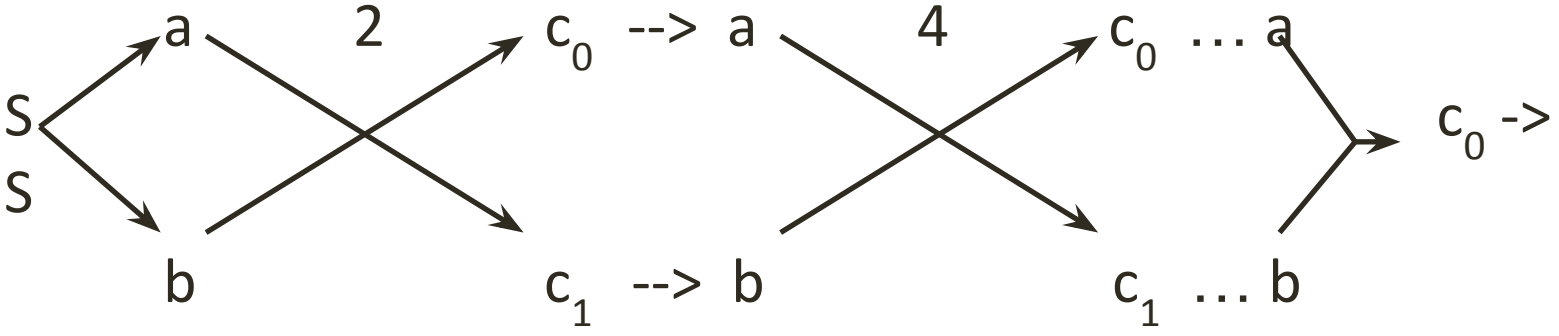
Пусть размер списка **S** = 2^k.

Список **S** расщепляем на два списка **a** и **b**.

Сливаем списки **a** и **b** с образованием **двойных серий**, то есть одиночные элементы сливаются в **упорядоченные пары**, которые записываются попеременно в очереди c_0 и c_1.

Переписываем очередь c_0 в список **a**, очередь c_1 – в список **b**.

Вновь сливаем списки **a** и **b** с образованием **серий длины 4 , эатем длины 8** и т. д.
На каждом шаге **размер серий увеличивается вдвое**.
Когда получим **одну серию длиной во весь список**, процесс сортировки завершен.



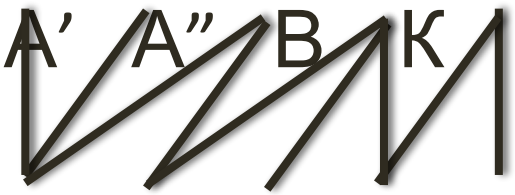
Замечание: Если размер списка не кратен степени двойки, то нужно учесть, что какие-то серии могут быть короче, чем ожидается.


S К У Р А’ П О В А” Е’ Л Е” Н А”’
a К Р П В Е’ Е” А”’
b У А’ О А” Л Н
a ←c0 К У О П Е’ Л А”’
b ←c1 А Р А” В Е” Н

$a \leftarrow c_0$ А’ К Р У Е’ Е” Л Н

$b \leftarrow c_1$ А” В О П А”’

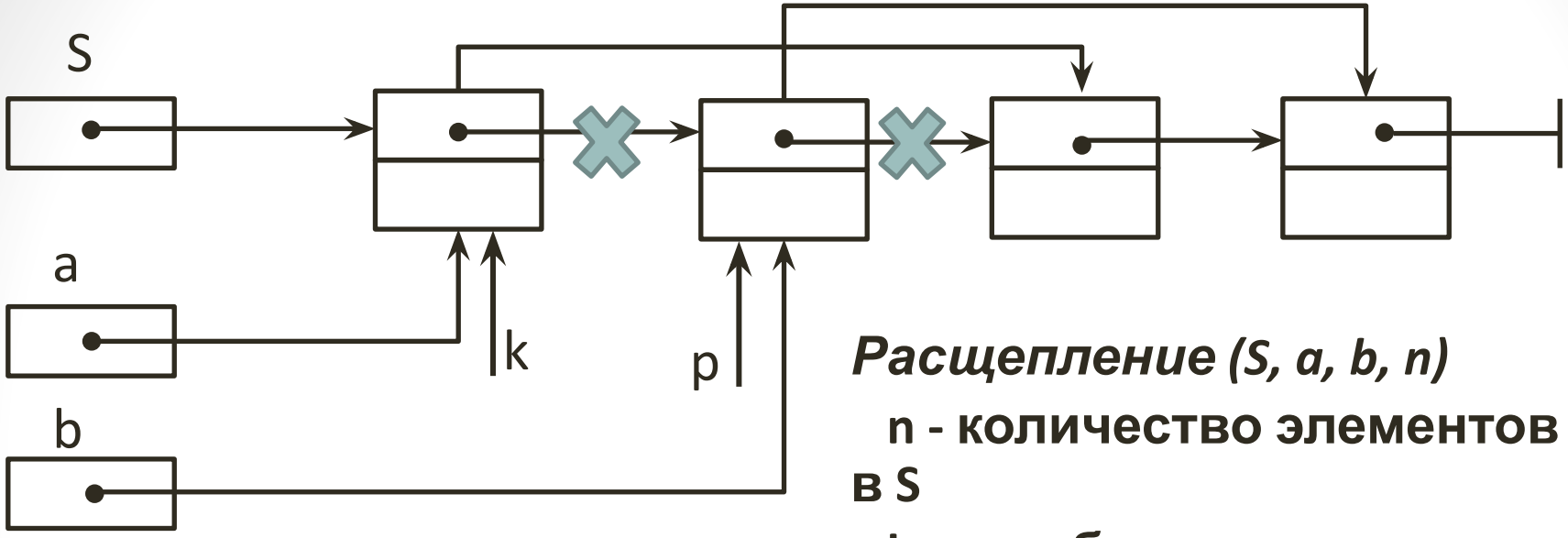
$a \leftarrow c_0$ А’ А” В К О П Р У



$b \leftarrow c_1$ А”’ Е’ Е” Л Н

S$\leftarrow c_0$ А’ А” А”’ В Е’ Е” К Л Н О П Р У

Алгоритм расщепления списка S



Расщепление (S, a, b, n)

n - количество элементов в S

k, p - рабочие указатели

a := S, b := S→Next, n := 1

k := a, p := b

DO (p ≠ NIL)

n := n+1

k→next := p→next

k := p

p := p→next

OD

Метод прямого слияния (MergeSort)

Обозначение переменных:

n – количество элементов в списке S
a, **b** – рабочие списки
c = (**c_0**, **c_1**) – массив из двух очередей
p – предполагаемый размер серии
q – фактический размер серии в списке **a**
r – фактический размер серии в списке **b**
m – текущее количество элементов в списках a и b
i – номер активной очереди

Метод прямого слияния (MergeSort)

```
< Расщепление (S, a, b, n) >
p := 1
        DO ( p < n )
            < инициализация очередей c0, c1 >
             i := 0, m := n
             DO ( m > 0 )
                IF ( m ≥ p ) q := p  ELSE  q := m  FI
                m := m – q
                IF ( m ≥ p ) r := p  ELSE  r := m  FI
                m := m – r
                < слияние(a, q, b, r, ci ) >
                i := 1– i
            OD
            a := c0 . Head,  b := c1. Head
            p := 2p
        OD
        c0 . Tail→next := NULL
        S := c0 . Head
```

Трудоемкость метода MergeSort

Трудоемкость сортировки следует из трудоемкости операции слияния серий.

На каждой итерации «большого» цикла производится ровно **n перемещений** элементов списков и **менее n сравнений** элементов.

Количество итераций равно $\lceil \log_2 n \rceil$

$$C < n \lceil \log_2 n \rceil$$

$$M = n \lceil \log_2 n \rceil + n$$

$$T = O(n \log_2 n)$$

Метод обеспечивает устойчивую сортировку.

При реализации алгоритма для массивов метод требует наличия второй рабочей копии массива.

При реализации алгоритма для списков копия массива не требуется.

Метод	Трудоемкость	Устойчивость	Зависимость от упорядоченности
ShellSort	$O(n^{1,2})$	Не устойчив	Зависит
HeapSort	$O(n \log_2 n)$	Не устойчив	Практически не зависит
QuickSort	$O(n \log_2 n)$	Не устойчив	Зависит
MergeSort	$O(n \log_2 n)$	Устойчив	?