Merge Sort

(I) Вспомним

$\Theta(N \log N)$

Пузырёком
Вставками
Выбор

$\Theta(N^2)$

(II) Ф-ия Merge.

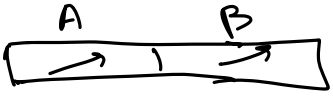
Два отсортированных массива, A и B

A 

B 

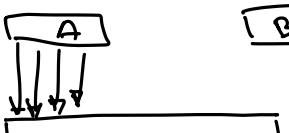
хотим получить в итоге один C 

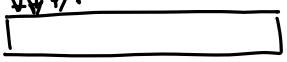
Тривиальн. алгоритм

C = 

⇒ отсортировали ⇒ ответ.
$\Theta(N^2)$

C 

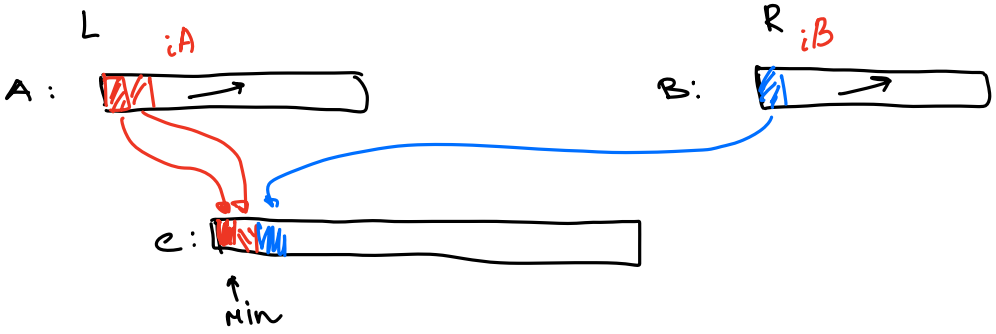


C = 

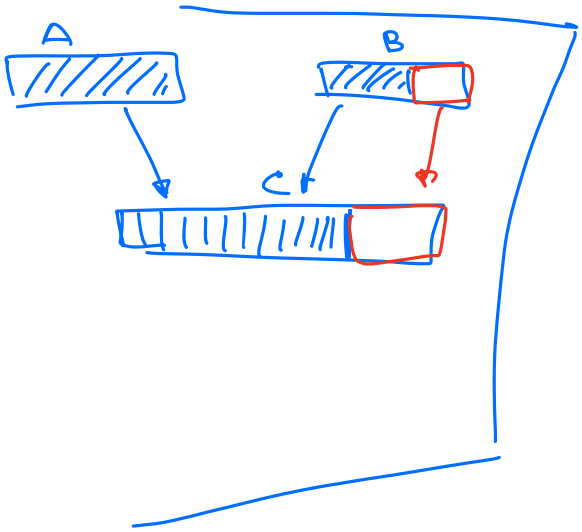
$\Theta(1) \cdot N = \Theta(N)$

$\Theta(N) + \Theta(N^2) =$
$= \Theta(N + N^2) =$
$= \Theta(N^2)$

Рабочий вариант



Алгоритм




```
Merge ( array A, array B)
{
    L = R = 0
    C = [ .... ]   // размер A.size + B.size
    while  L < A.size  and  R < B.size :
    {
        если  A[L] < B[R]
            ↳ C[L+R] = A[L]
              L = L+1
        иначе
            ↳ C[L+R] = B[R]
              R = R+1
    }
    while  L < A.size :
    {
        C[L+R] = A[L]
        L = L+1
    }
    while  R < B.size:
        C[L+R] = B[R]
        R = R+1
    }
    return C
}
```

Асимптотика:

по времени

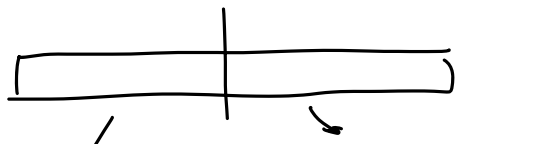
$$T(N) = \underline{O}(\overbrace{A.size + B.size}^{N}) = \underline{O}(N)$$

по памяти

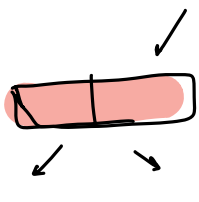
$$M(N) = \underline{O}(N)$$

III Дивайд Merge Sort

N-размерный array:



$\frac{N}{2}$



$\frac{N}{4}$

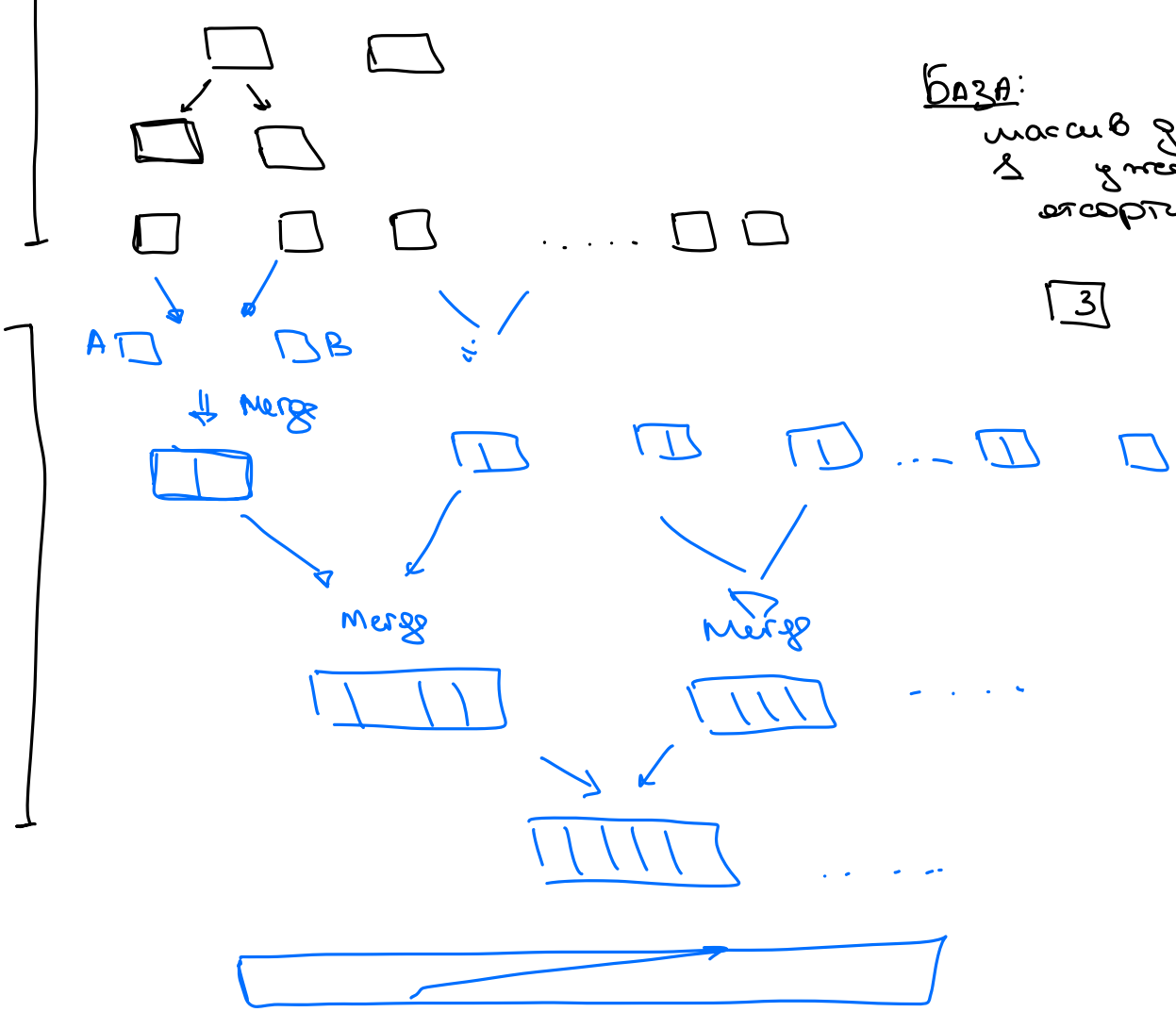
$\frac{N}{8}$

$\dots$

$\frac{N}{2^k} = 1$

$N = 2^k$

$k = \log_2 N$



База:
массив длины 1 уже отсортирован

$\boxed{3}$

Высота: $2\log_2 N$

Память: $\Theta(N)$

Время: $2\log_2 N \cdot \Theta(N) =$
$= \Theta(2\log_2 N \cdot N) =$
$= \Theta(N \log_2 N) =$
$= \Theta(N \log N)$

Алгоритм:

```
MergeSort ( array)
{
    N=  array.size
    если  N = 1
      ↳ return array.
    * A = array [0... N//2]
    * B = array [N//2 .... N]
      A = MergeSort (A)
      B = MergeSort (B)
```

return Merge (A, B)

}

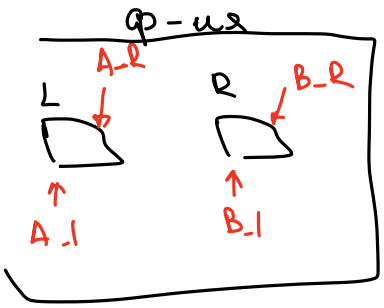
(IV) Упрощение Merge Sort

1. Merge Sort (array, left, right)

array

Merge Sort (array, left, $\frac{left+right}{2}$)

Merge Sort (array, $\frac{left+right}{2}$, right)



2. Создать буферный массив.

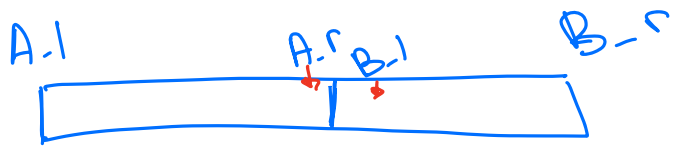
Merge c = [...]

```
buffer = [....]  // размер N

                                  A          B
Merge ( array, A_l, A_r, B_L, B_R, buffer):
    L = A_l
    R = B_l
    i = 0
    while  L < A_r  and  R < B_r:
    {
        если  array[L] < array[R]:
                buffer[i] = array[L]
                i = i+1
                L = L+1
        иначе
                buffer[i] = array[R]
                i = i+1
                R = R+1
    }

    while  L < A_r
    {      ↳ buffer[i] = A[L]
```

```
        i = i+1
        L = L+1
    }
    while    R < B_r
    }        buffer[i] = B[R]
             i = i+1
             R = R+1
    }
    for  id = A_l .... B_r :
    ↳        array[id] = buffer[id - A_l]
}
```




```
MergeSort( array, left, Right)   buffer
{
    Base   right - left ≤ 1
      ↳ return
    M = ⌊(left + right)/2⌋
    mergeSort( array, left, M)   buffer
    mergeSort( array, M+1, right)   buffer
    merge( array, left, M, M+1, right)   buffer.
}
```