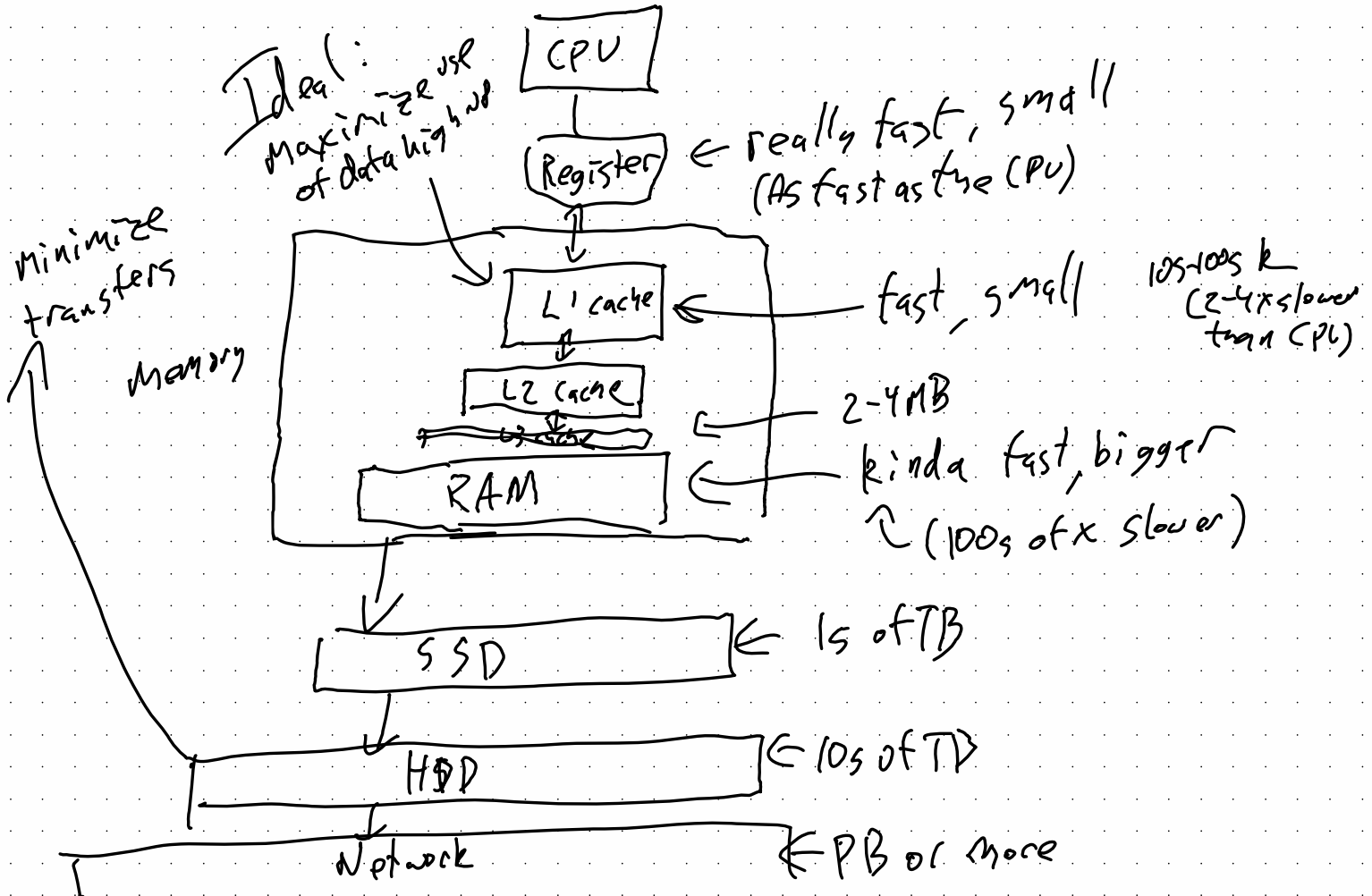
Memory access = $O(1)$

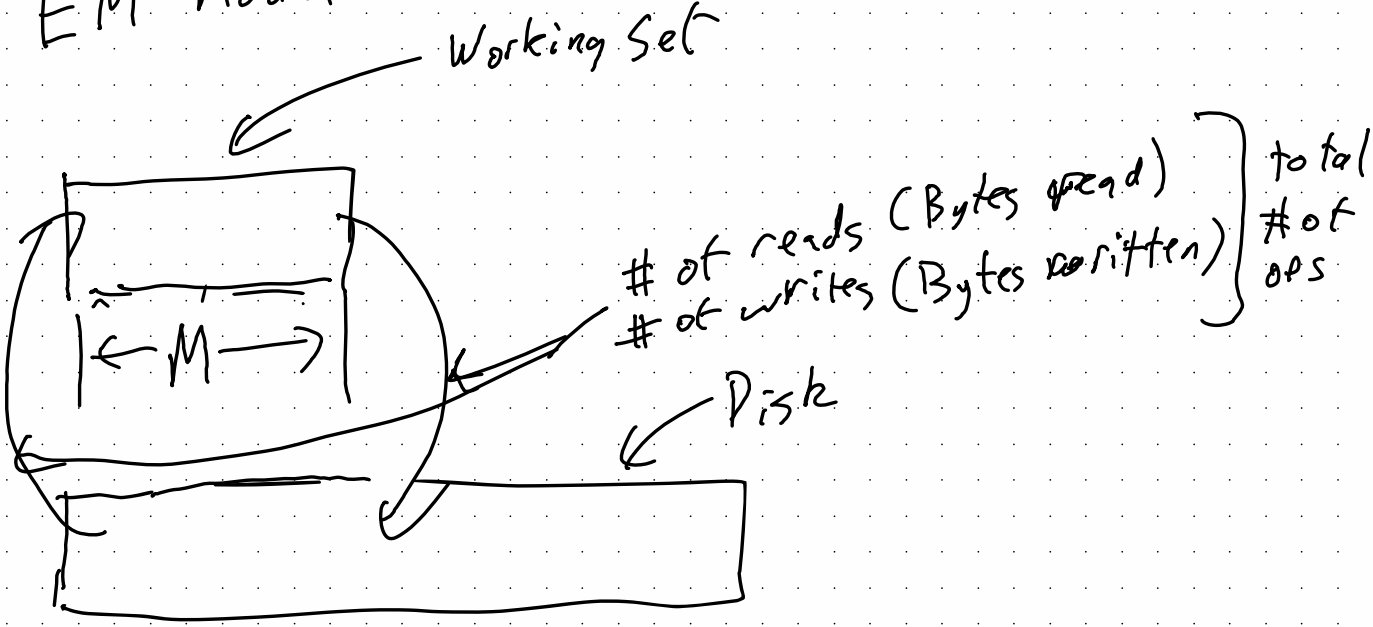
↑ RAM model of computation

vs

External Memory (EM)

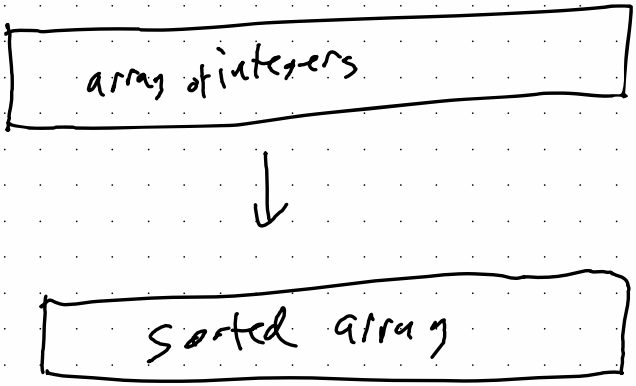

Ideal: maximize use of data high
CPU
Register
really fast, small
(As fast as the CPU)
Minimize transfers
Memory
L1 cache
fast, small
10s-100s k
(2-4x slower than CPU)
L2 cache
2-4MB
RAM
kinda fast, bigger
(100s of x slower)
SSD
1s of TB
HDD
10s of TB
Network
PB or more

EM Model

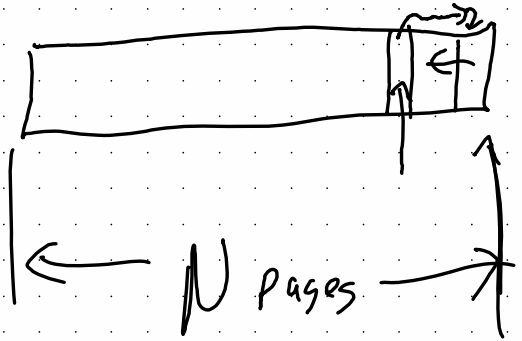


Measures

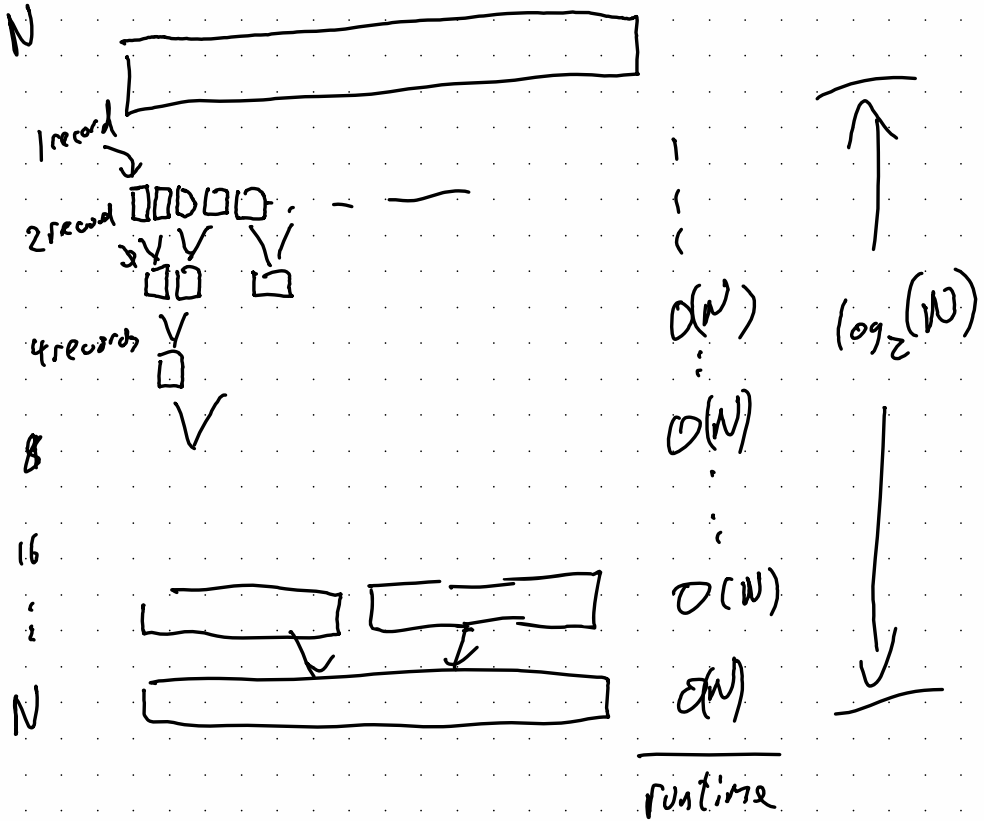
- Runtime complexity
- Memory Complexity: |The biggest working set|
- IO complexity: Total # of disk accesses
  - ↳ # of random vs # of seq
  - ↳ # of reads vs # writes


array of integers
sorted array

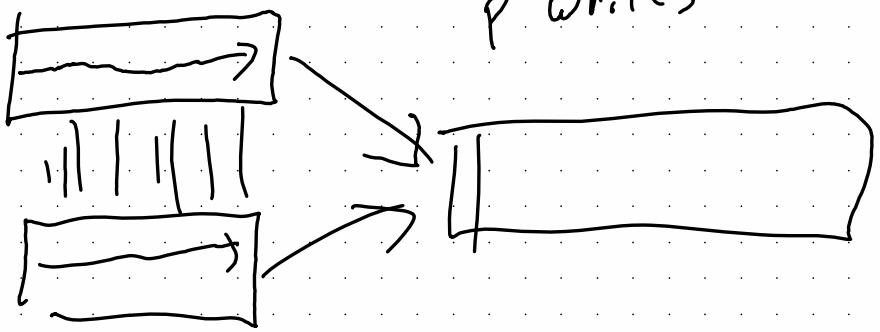
Bubble sort



Merge Sort



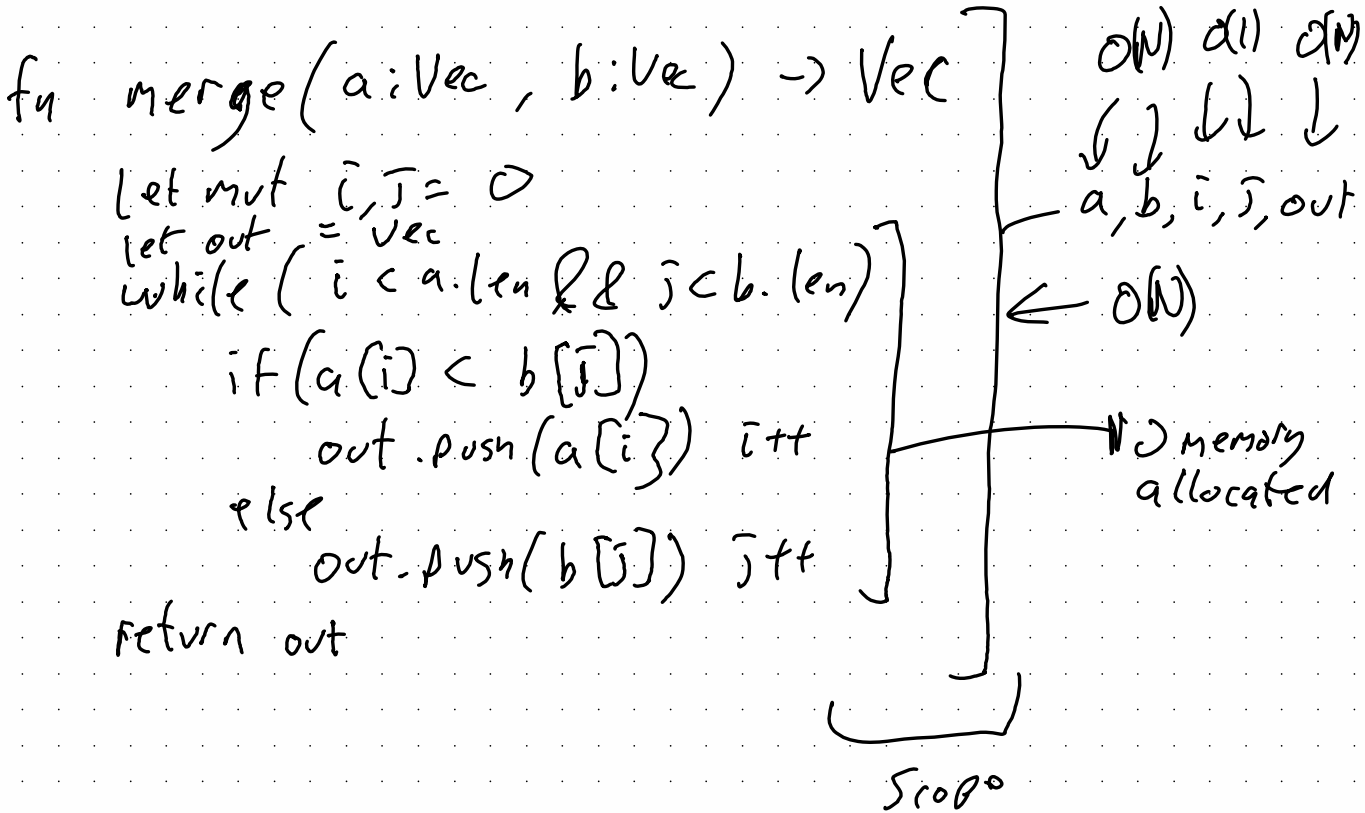
$\frac{P}{2}$ pages per input



? reads
P writes
} IO : O(P)
≃ O(N)

↑ Storing only 2 records at a time
memory: O(1)

One pass algorithm
↑ Only look at each record once


fn merge(a:Vec, b:Vec) -> Vec
let mut i,j = 0
let out = Vec
while (i < a.len && j < b.len)
if(a[i] < b[j])
out.push(a[i]) i++
else
out.push(b[j]) j++
return out
O(N) O(1) O(N)
a, b, i, j, out
O(N)
memory allocated
Scope

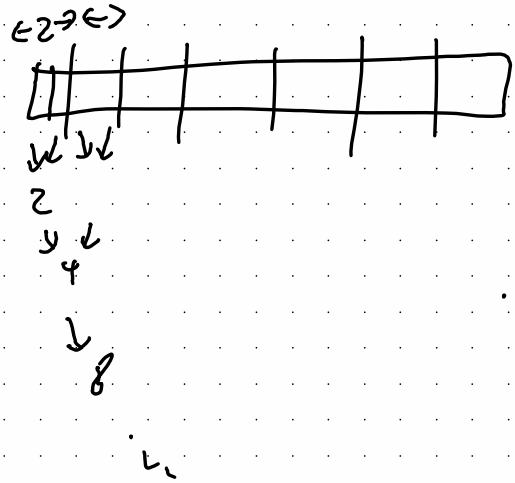
read chunk
write chunk

```
fn merge( a: disk , b: disk ) -> c: disk
    let mut i, j, k = 0
    let a_val, b_val = a[i] , b[j]        } O(N) IOs
    while ( i < a_len && j < b_len )      } O(N) IOs
        if a_val < b_val                  } O(1) IOs
            out[k] = a_val; i++; k++; a_val = a[i]
        else
            out[k] = b_val; j++; k++; b_val = b[j]
```

i, j, k, a_val, b_val ↑ O(1)

$O(1) + O(N) = O(N)$ Total IOs

	Runtime	Memory	IOs
Merge_on_disk	O(N)	O(1)	O(N)



```
fn merge ( input: disk )
    let size = 1
    while(size < input.len/2)
        let i = 0
        while(i                < input.len)
            merge_on_disk( input[i,i+size], input[i+size, i+2*size])
            i += 2*size
        size *= 2
```

O(log(N)) times

No IO

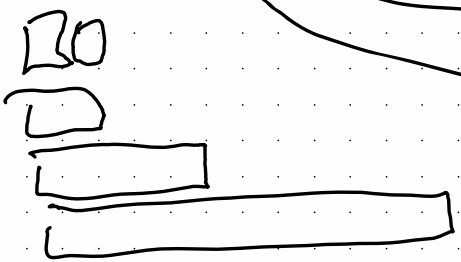
No IO

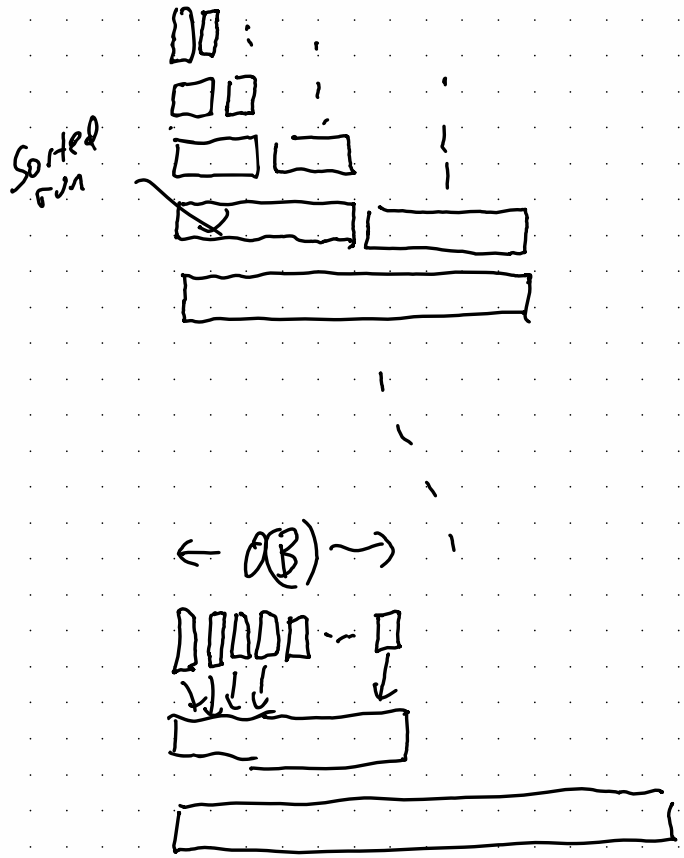
O(size) IOs

repeat $\frac{N}{size}$ times

O(N) total IOs

O(N log(N)) total IOs





O(B) memory

Buffering skips $O(\log_2 B)$ steps

1
2
4
8
16
32
B
N

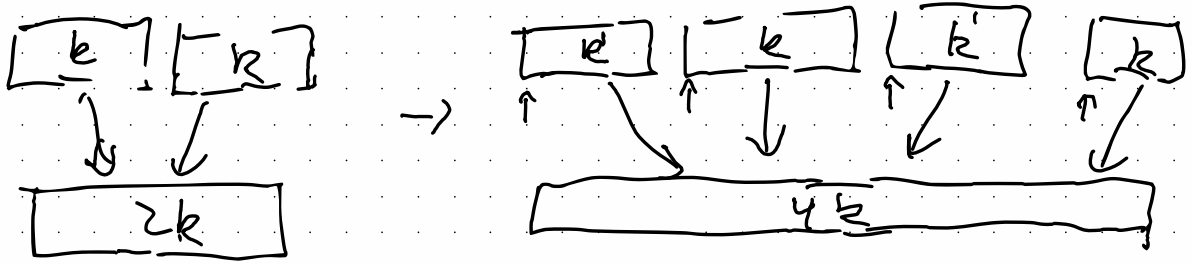
$O(\log_2 N) - O(\log_2 B)$

$O(\log_2 N - \log_2 B)$

$O(\log_2 \frac{N}{B})$

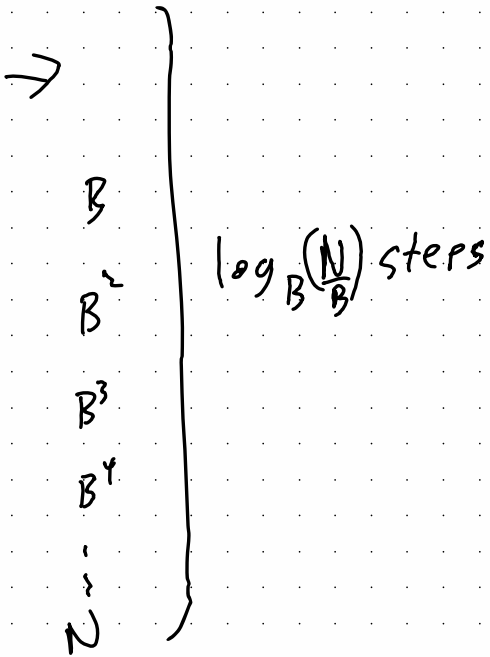
IOs

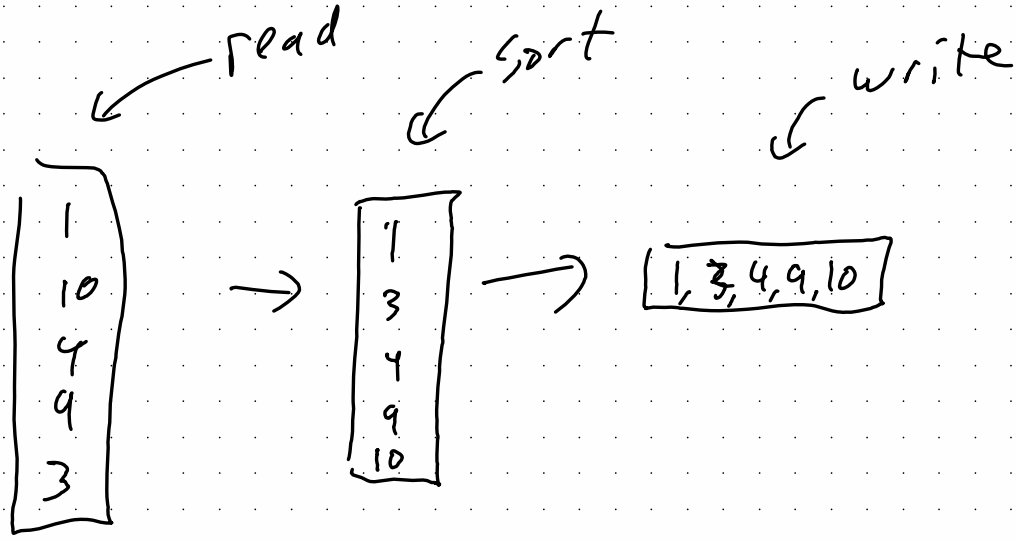
Regular Sort-Merge



B sorted runs

↳ New run of size BN





5
3