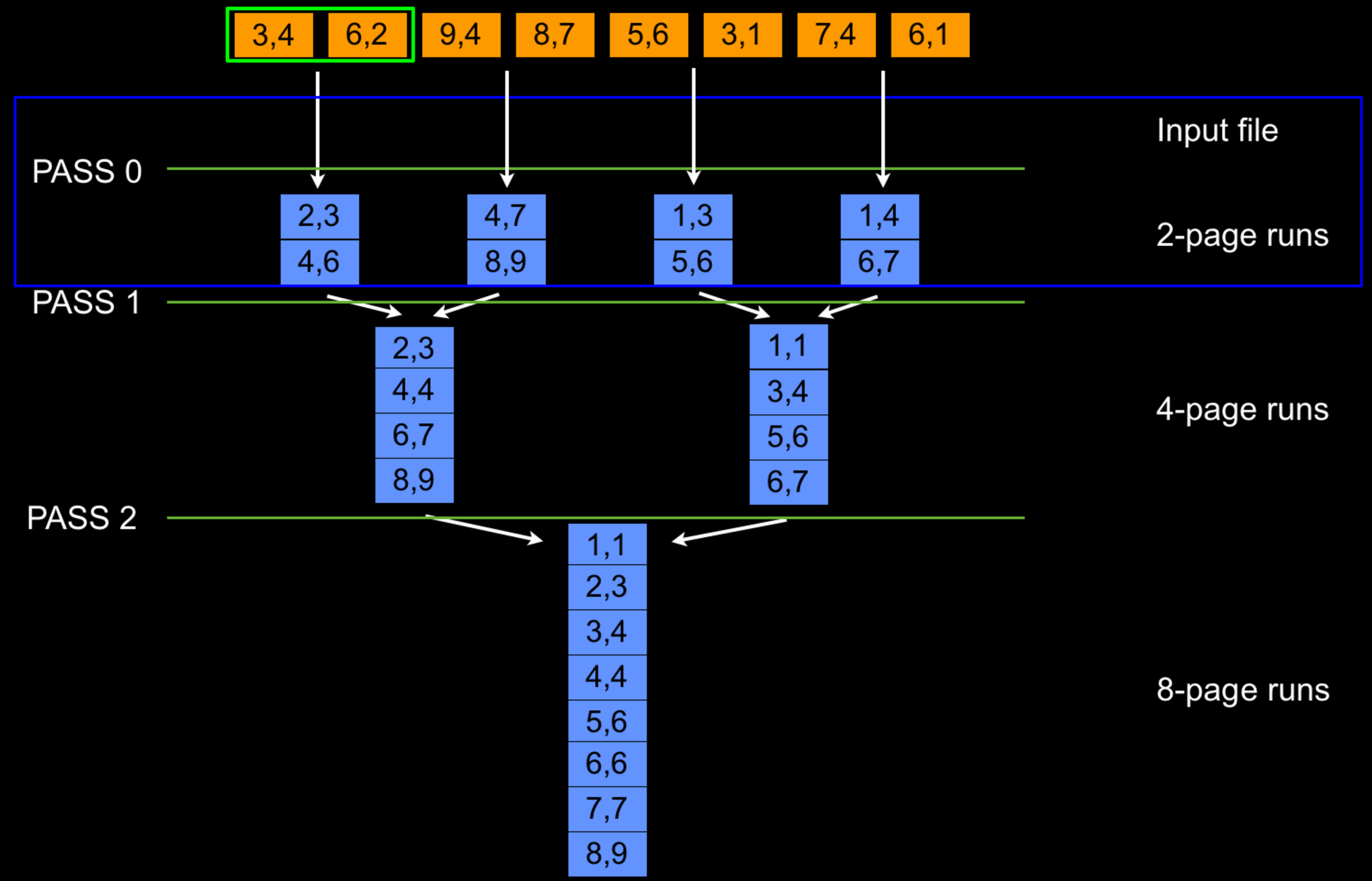
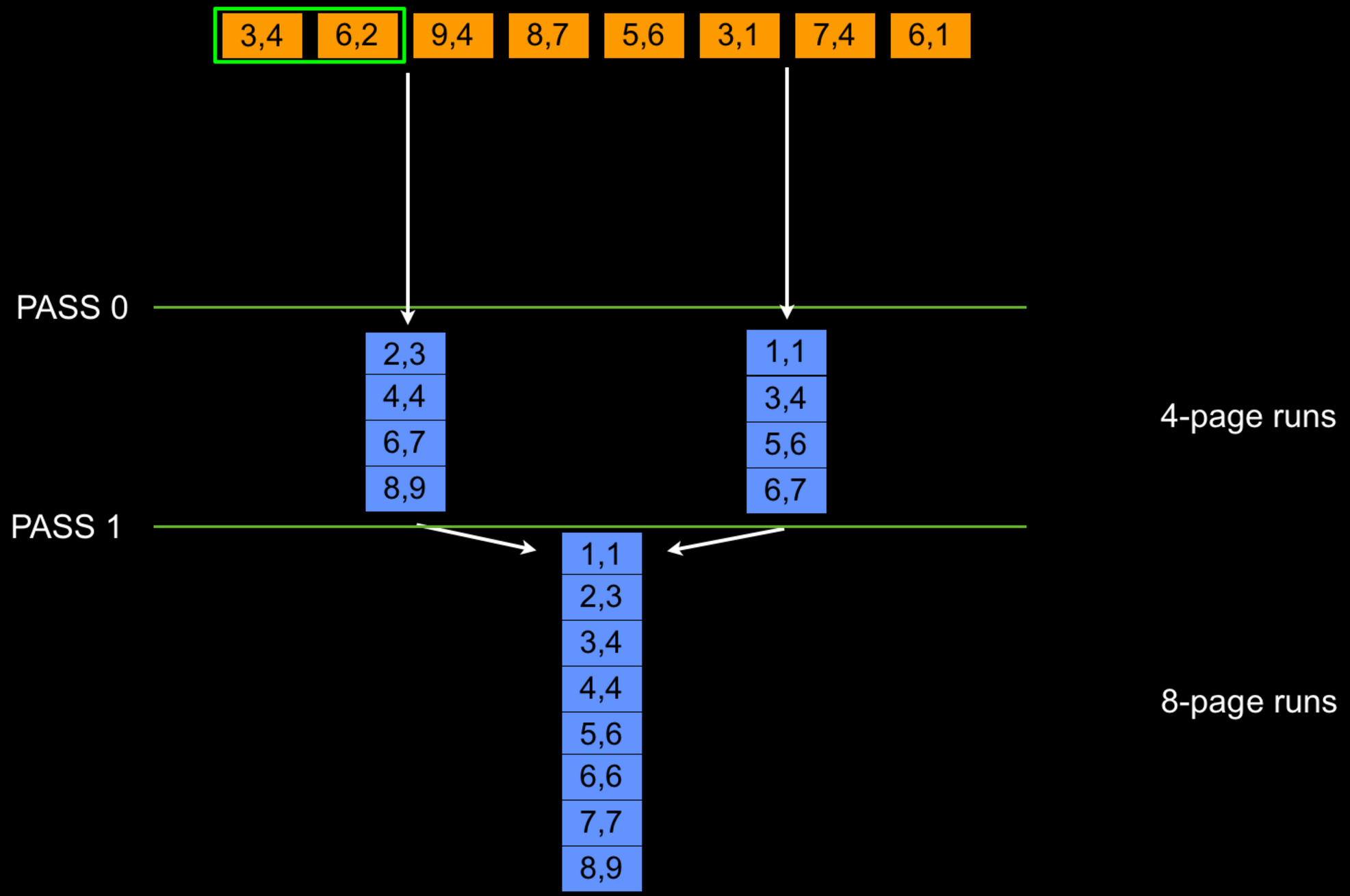
Two Main Memory Pages for Run Generation with Quicksort



Two Main Memory Pages for Run Generation with Replacement Selection



Replacement Selection Example

M = 4

Output run R'

heap

Input R

30	20	10	40	25	73	16	26	33	50	31

array

list

$k \leq M \rightarrow$ heap

$M-k \rightarrow$ list

$|heap| + |list| \leq M$

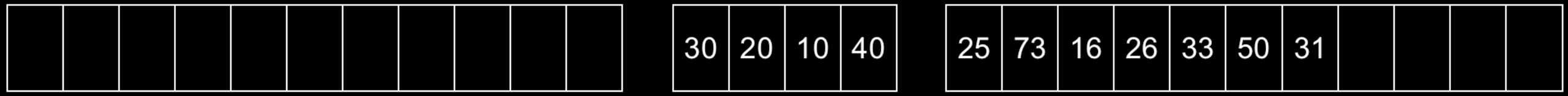
Replacement Selection Example

M = 4

Output run R’

heap

Input R



list

$O(n \log n)$ tuple-wise insert

$O(n)$ bulkload

|heap| + |list| ≤ M

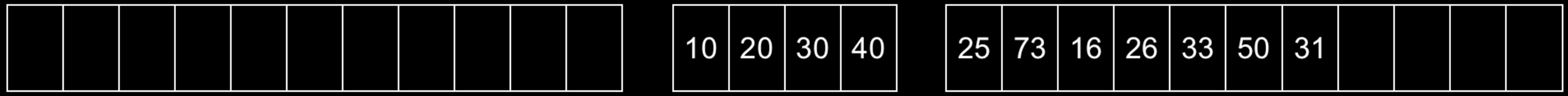
Replacement Selection Example

M = 4

Output run R’

heap

Input R



list

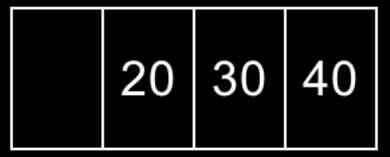
|heap| + |list| ≤ M

Replacement Selection Example

M = 4

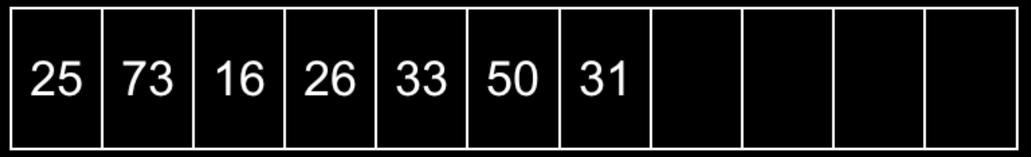
Output run R’

heap



list

Input R



$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

										10

heap

20	25	30	40

Input R

73	16	26	33	50	31					

list

$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

									10	20

heap

	25	30	40

Input R

73	16	26	33	50	31					

list

$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

									10	20

heap

25	30	40	73

Input R

16	26	33	50	31						

list

$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

								10	20	25

heap

	30	40	73

Input R

16	26	33	50	31						

list

|heap| + |list| ≤ M

Replacement Selection Example

M = 4

Output run R’

								10	20	25

heap

	30	40	73

Input R

26	33	50	31							

16			

list

$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

							10	20	25	30

heap

		40	73

Input R

26	33	50	31							

16			

list

|heap| + |list| ≤ M

Replacement Selection Example

M = 4

Output run R’

							10	20	25	30

heap

		40	73

16	26		

list

Input R

33	50	31								

$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

						10	20	25	30	40

heap

			73

16	26		

list

Input R

33	50	31								

$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

						10	20	25	30	40

heap

			73

16	26	33	

list

Input R

50	31									

$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

					10	20	25	30	40	73

heap

16	26	33	

list

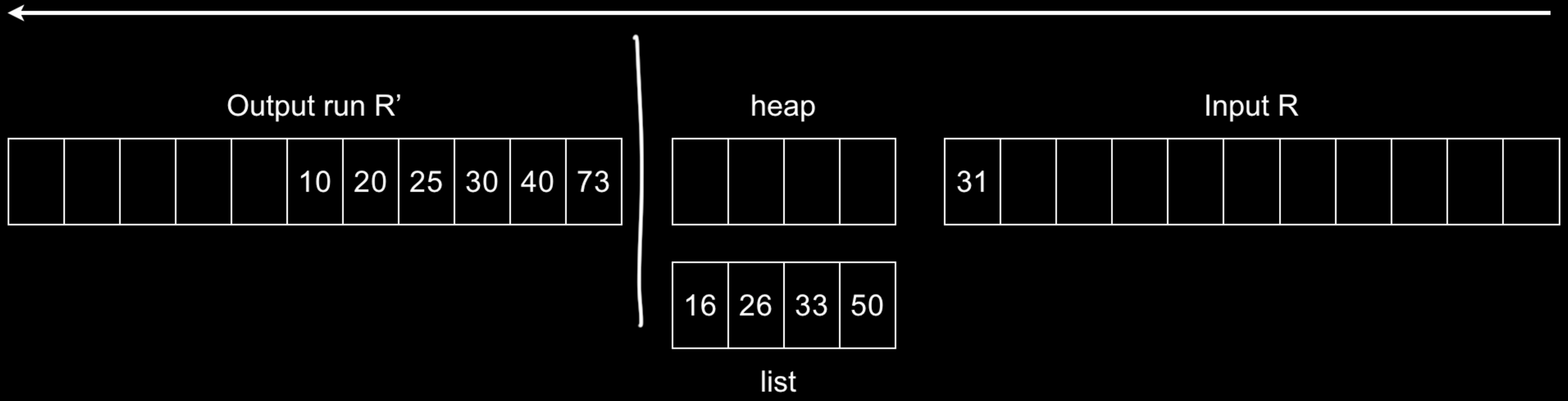
Input R

50	31									

$|heap| + |list| \leq M$

Replacement Selection Example

M = 4



$|heap| + |list| \leq M$

Replacement Selection Example

M = 4

Output run R’

					10	20	25	30	40	73

heap

16	26	33	50

Input R

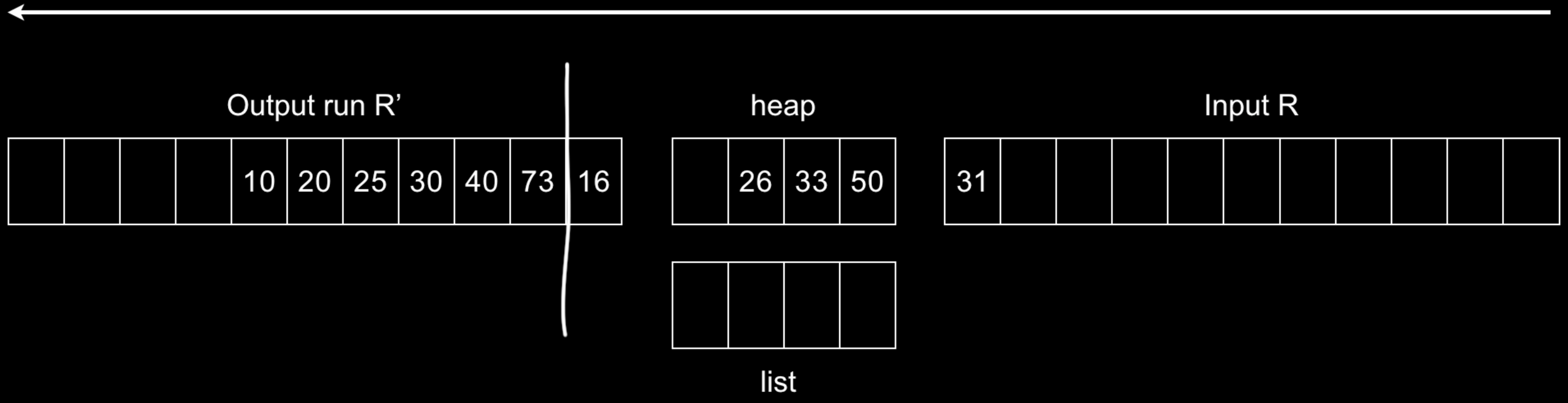
31										

list

$|heap| + |list| \leq M$

Replacement Selection Example

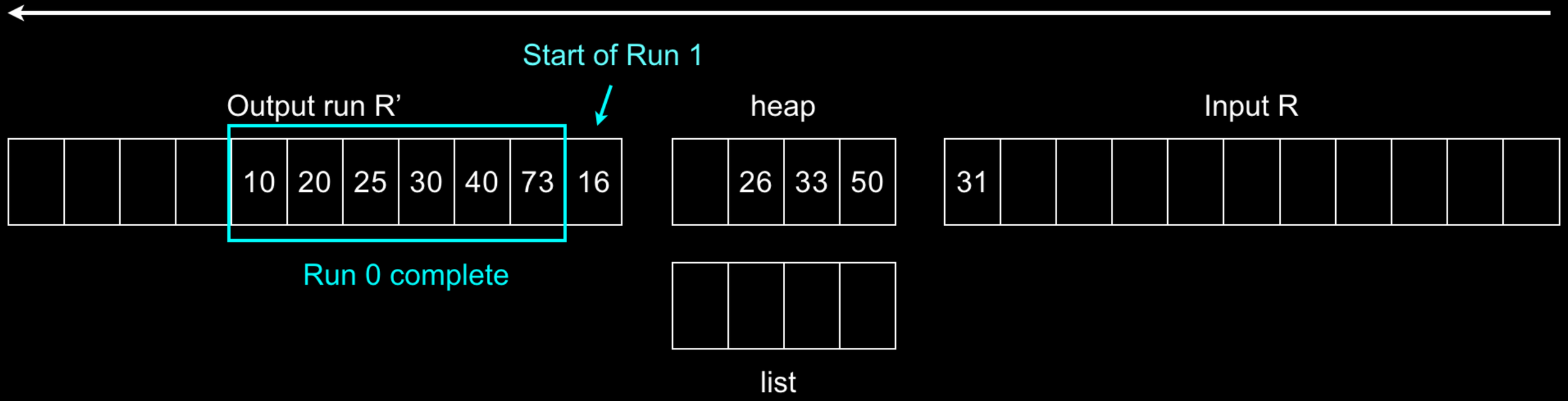
M = 4



|heap| + |list| ≤ M

Replacement Selection Example

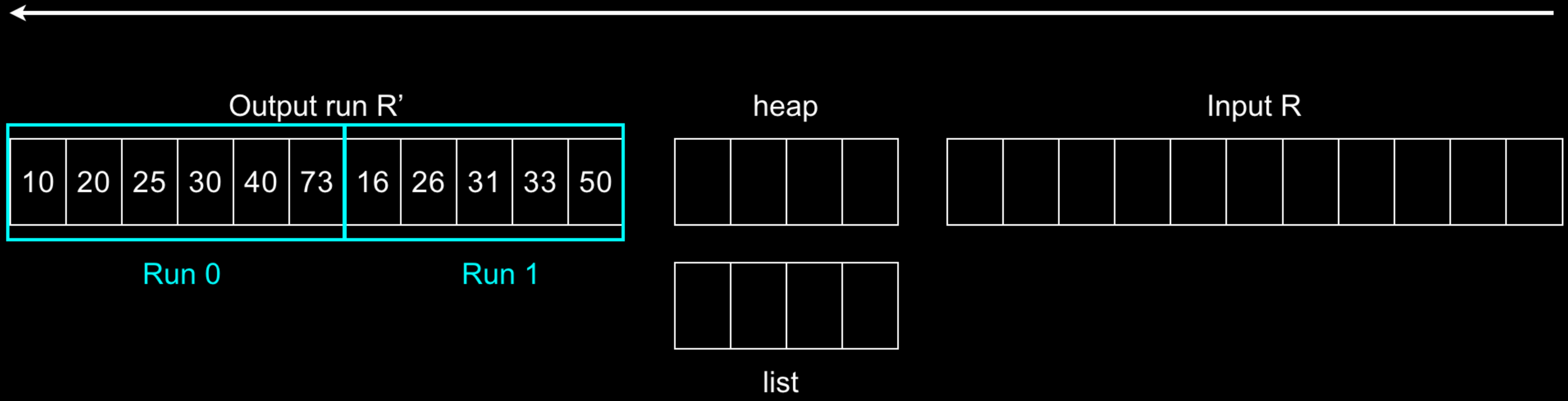
M = 4



$|heap| + |list| \leq M$

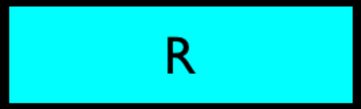
Replacement Selection Example

M = 4



|heap| + |list| ≤ M

Replacement Selection

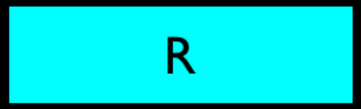


M := # elements in main memory
R := input queue
R‘ := output queue

```
ReplacementSelection( R ):
    Buffer B = read( R, M );                    //read M records from queue R into buffer B
    Heap heap = heapify( B );                   //bulkload the data of buffer B into a heap
    List list = new List();                     //create empty list
    do {                                        //do while heap not empty
        While NOT heap.isEmpty() {              //while heap contains elements
            Tuple r‘ = heap.pop();              //remove top element (smallest on the heap)
            R‘.append( r‘ );                    //append that element to output queue R‘
            If NOT R.isEmpty():                 //i.e. only if more elements in input R available
                Tuple next = read( R, 1 );      //read M records from queue R into variable next
                If next ≥ r‘:                   //next ≥ ‘last output‘ r‘? (Note: what if next == r‘?)
                    heap.push( next );          //put it on the heap
                Else:                           //i.e. next < ‘last output‘ r‘!
                    list.append( next );        //append to list, i.e. need to treat this later
        }                                       //end of while-loop
        heap = heapify( list );                 //bulkload the data of the list into a heap
        list = new List();                      //create a new empty list
```

1 array

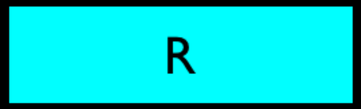
Replacement Selection



M := # elements in main memory
R := input queue
R‘ := output queue

```
ReplacementSelection( R ):
    Buffer B = read( R, M );                    //read M records from queue R into buffer B
    Heap heap = heapify( B );                   //bulkload the data of buffer B into a heap
    List list = new List();                     //create empty list
    do {                                        //do while heap not empty
        While NOT heap.isEmpty() {              //while heap contains elements
            Tuple r‘ = heap.pop();              //remove top element (smallest on the heap)
            R‘.append( r‘ );                    //append that element to output queue R‘
            If NOT R.isEmpty():                 //i.e. only if more elements in input R available
                Tuple next = read( R, 1 );      //read M records from queue R into variable next
                If next ≥ r‘:                   //next ≥ ‘last output‘ r‘? (Note: what if next == r‘?)
                    heap.push( next );          //put it on the heap
                Else:                           //i.e. next < ‘last output‘ r‘!
                    list.append( next );        //append to list, i.e. need to treat this later
        }                                       //end of while-loop
        heap = heapify( list );                 //bulkload the data of the list into a heap
        list = new List();                      //create a new empty list
    } While NOT heap.isEmpty()                  //continue until heap is empty
```

Replacement Selection



M := # elements in main memory
R := input queue
R‘ := output queue

```
ReplacementSelection( R ):
    Buffer B = read( R, M );                    //read M records from queue R into buffer B
    Heap heap = heapify( B );                   //bulkload the data of buffer B into a heap
    List list = new List();                     //create empty list
    do {                                        //do while heap not empty
        While NOT heap.isEmpty() {              //while heap contains elements
            Tuple r‘ = heap.pop();              //remove top element (smallest on the heap)
            R‘.append( r‘ );                    //append that element to output queue R‘
            If NOT R.isEmpty():                 //i.e. only if more elements in input R available
                Tuple next = read( R, 1 );      //read M records from queue R into variable next
                If next ≥ r‘:                   //next ≥ ‘last output‘ r‘? (Note: what if next == r‘?)
                    heap.push( next );          //put it on the heap
                Else:                           //i.e. next < ‘last output‘ r‘!
                    list.append( next );        //append to list, i.e. need to treat this later
        }                                       //end of while-loop
        heap = heapify( list );                 //bulkload the data of the list into a heap
        list = new List();                      //create a new empty list
    } While NOT heap.isEmpty()                  //continue until heap is empty
```