Counting Inversion

Counting Inversion

- Given ‘n’ numbers $a_1, a_2, . . a_n$
- Two indices $i < j$ form an inversion if $a_i > a_j$
- for example

Indices	**1**	**2**	**3**	**4**
Elements	10	7	9	12

- Inversion
  - 10-7 and 10-9
- Simple implementation takes (brute force) $O(n^2)$

Counting Inversion: Divide & Conquer

- Count the number of inversion

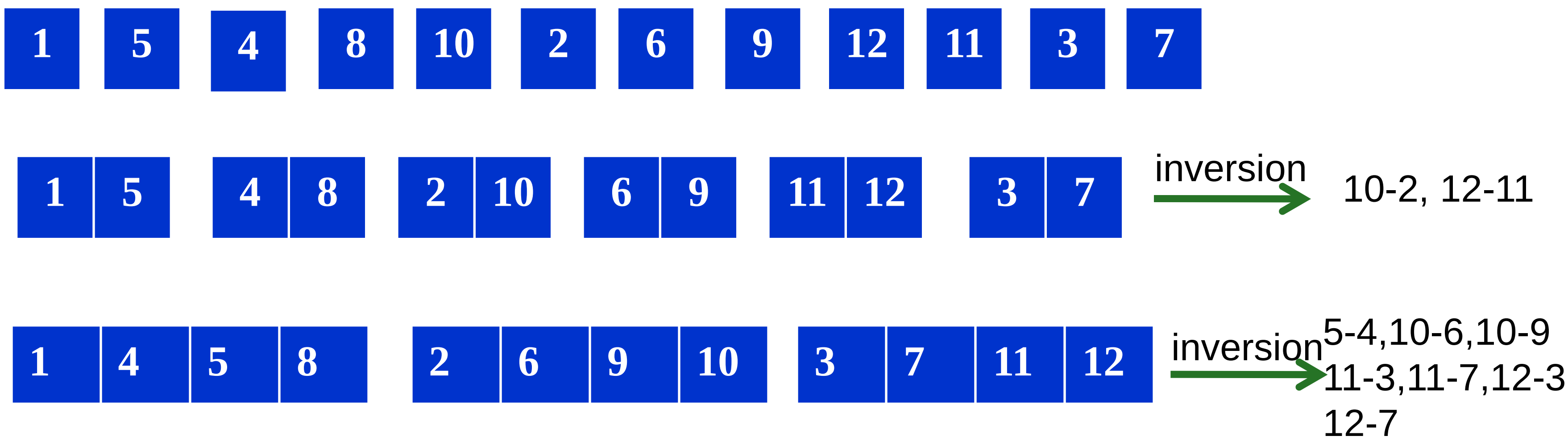
1	5	4	8	10	2	6	9	12	11	3	7

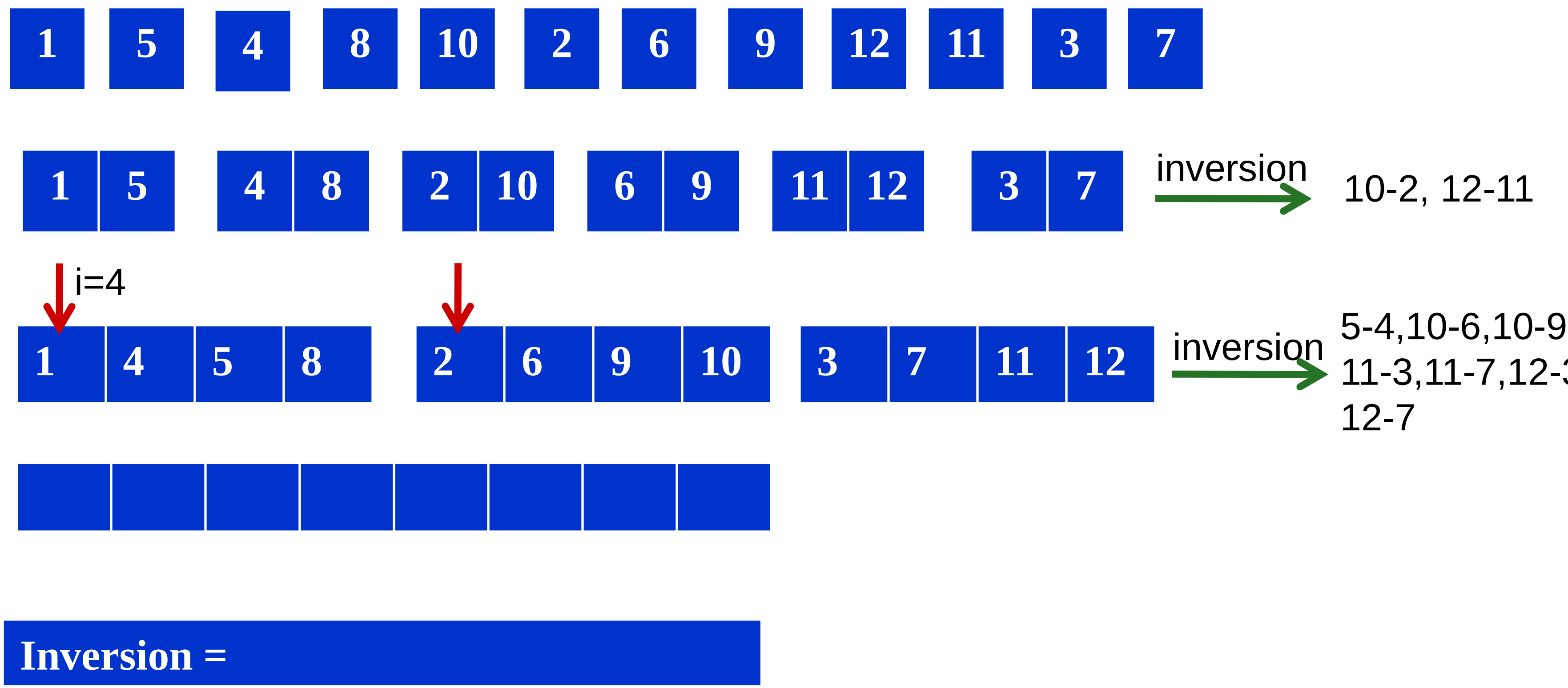
1	5	4	8	10	2		6	9	12	11	3	7

1	5	4		8	10	2		6	9	12		11	3	7

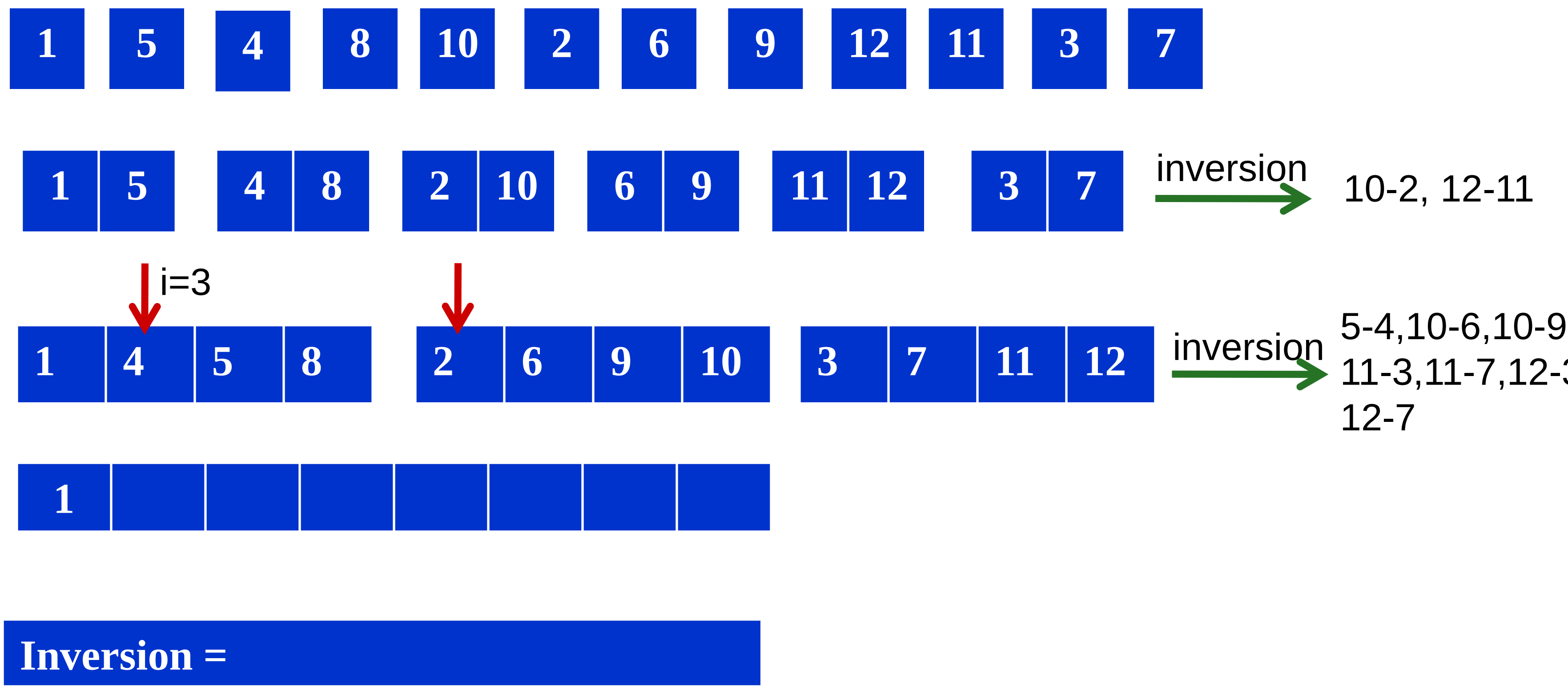
1		5	4		8		10	2		6		9	12		11		3	7

1	5	4	8	10	2	6	9	12	11	3	7

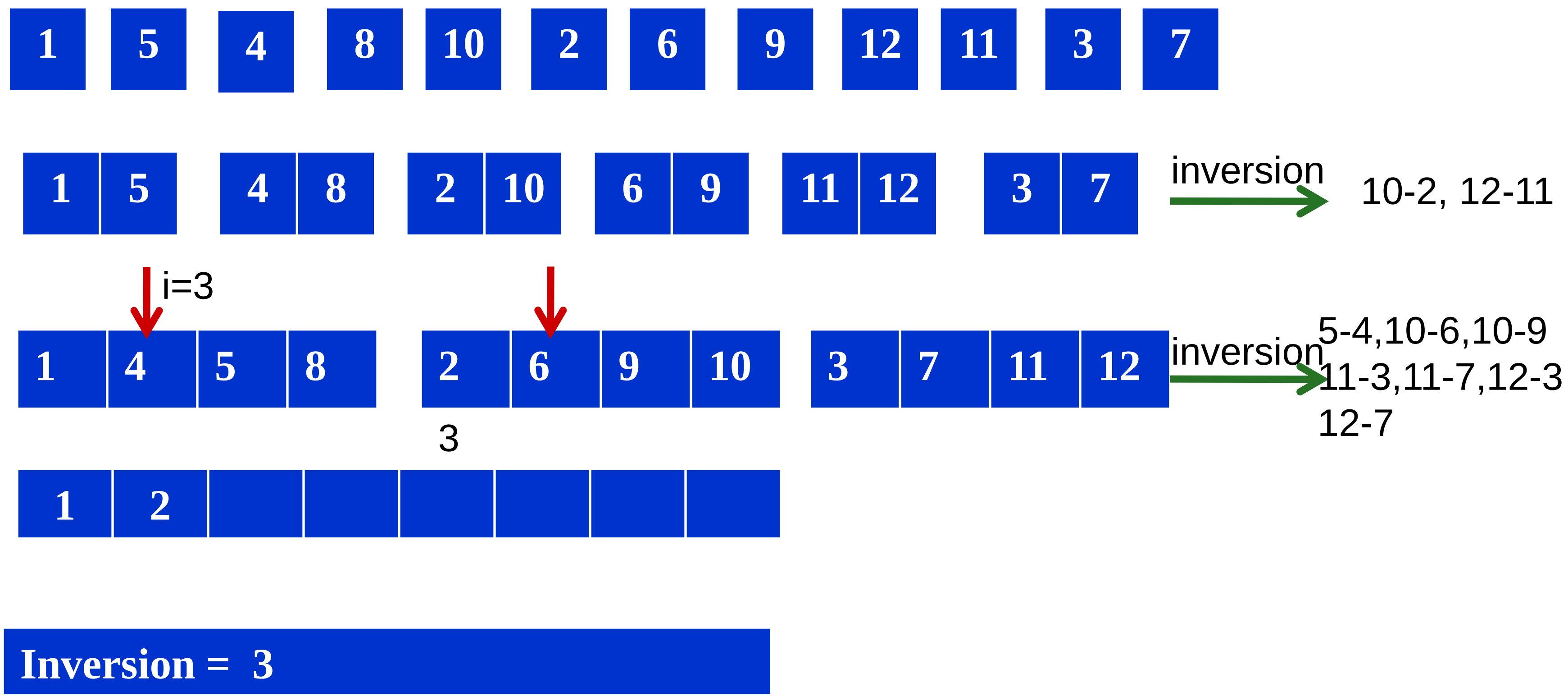

1 5 4 8 10 2 6 9 12 11 3 7
1 5 4 8 2 10 6 9 11 12 3 7
inversion
10-2, 12-11
1 4 5 8 2 6 9 10 3 7 11 12
inversion
5-4,10-6,10-9
11-3,11-7,12-3
12-7



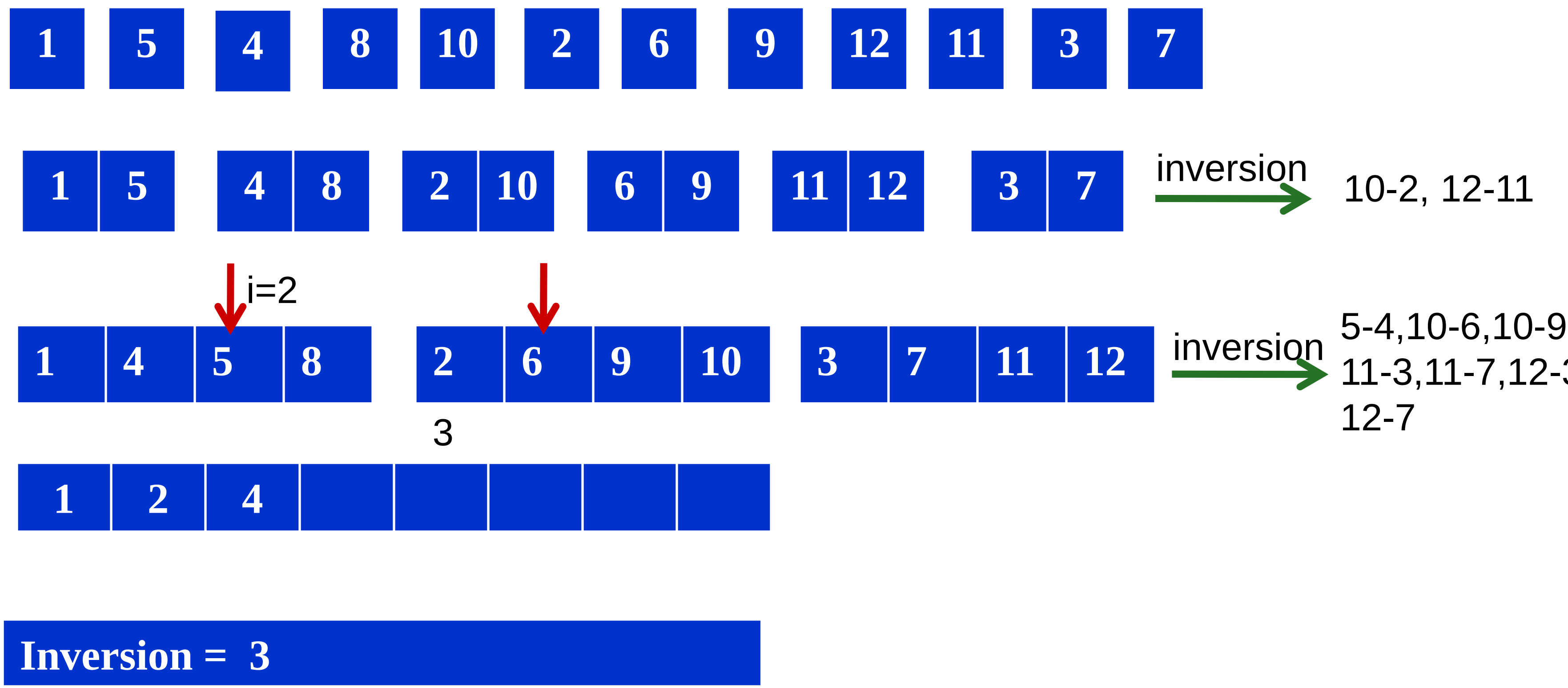
1 5 4 8 10 2 6 9 12 11 3 7
1 5 4 8 2 10 6 9 11 12 3 7
inversion
10-2, 12-11
i=4
1 4 5 8 2 6 9 10 3 7 11 12
inversion
5-4,10-6,10-9
11-3,11-7,12-
12-7
Inversion =



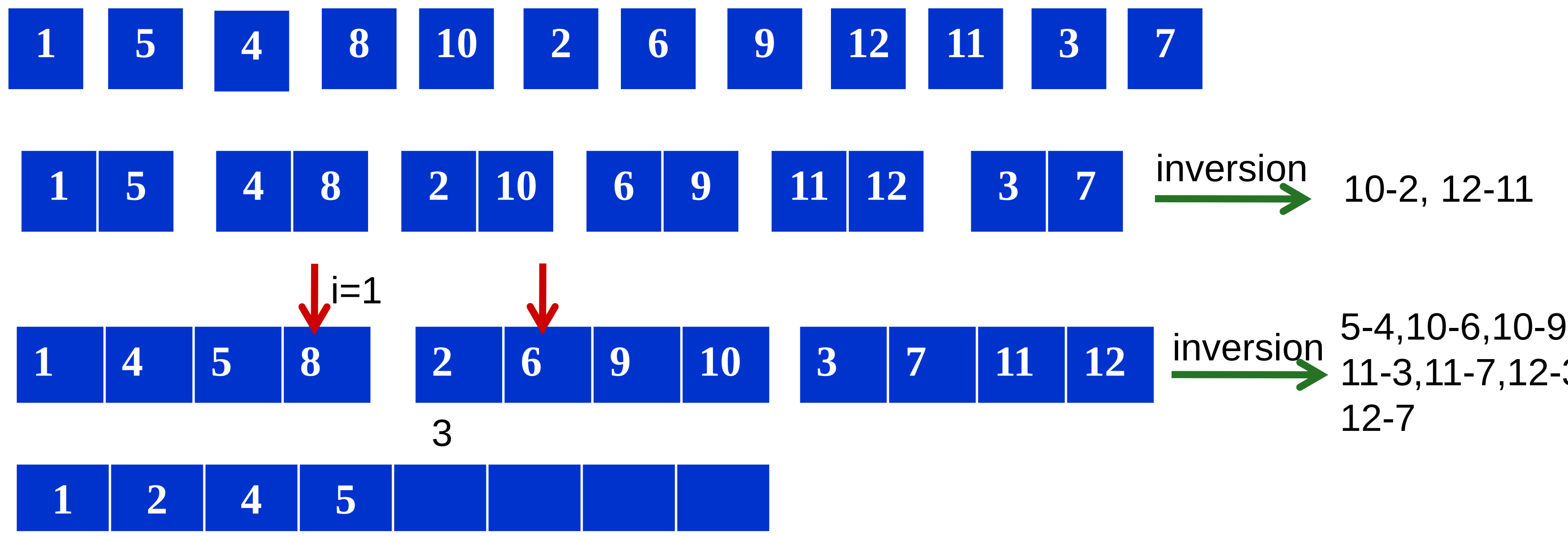
1 5 4 8 10 2 6 9 12 11 3 7
1 5 4 8 2 10 6 9 11 12 3 7
inversion
10-2, 12-11
i=3
1 4 5 8 2 6 9 10 3 7 11 12
inversion
5-4,10-6,10-9
11-3,11-7,12-3
12-7
1
Inversion =



1 5 4 8 10 2 6 9 12 11 3 7
1 5 4 8 2 10 6 9 11 12 3 7
inversion
10-2, 12-11
i=3
1 4 5 8 2 6 9 10 3 7 11 12
inversion
5-4,10-6,10-9
11-3,11-7,12-3
12-7
3
1 2
Inversion = 3

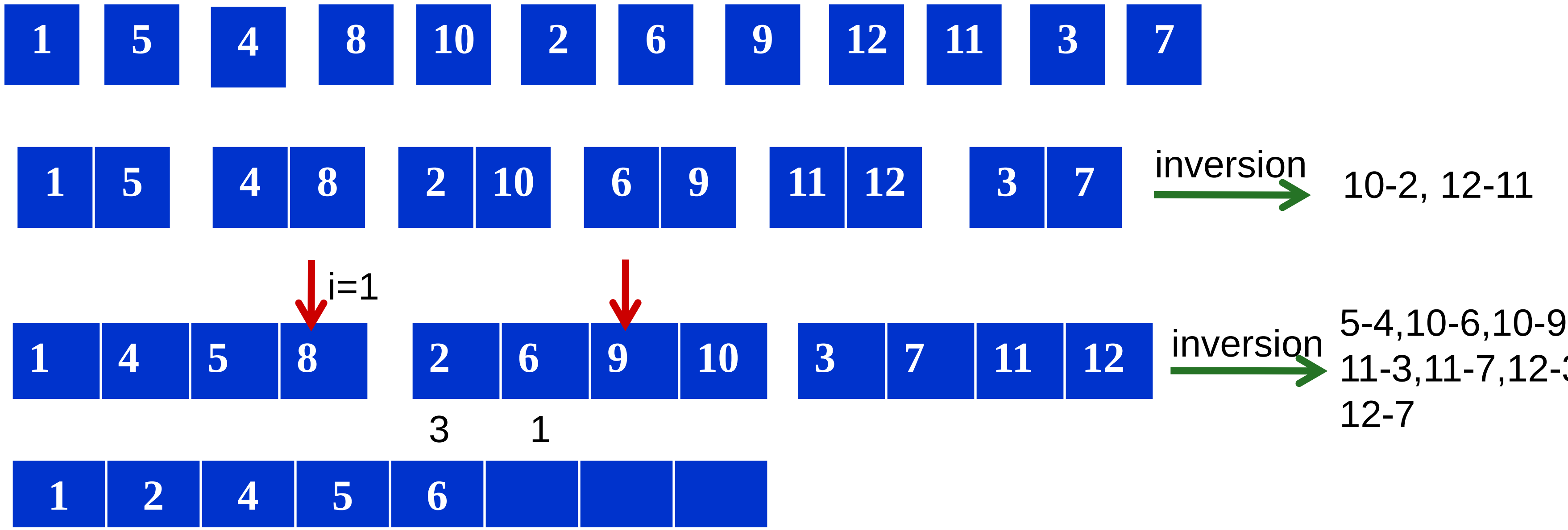


1 5 4 8 10 2 6 9 12 11 3 7
1 5 4 8 2 10 6 9 11 12 3 7
inversion
10-2, 12-11
i=2
1 4 5 8 2 6 9 10 3 7 11 12
inversion
5-4,10-6,10-9
11-3,11-7,12-
12-7
3
1 2 4
Inversion = 3

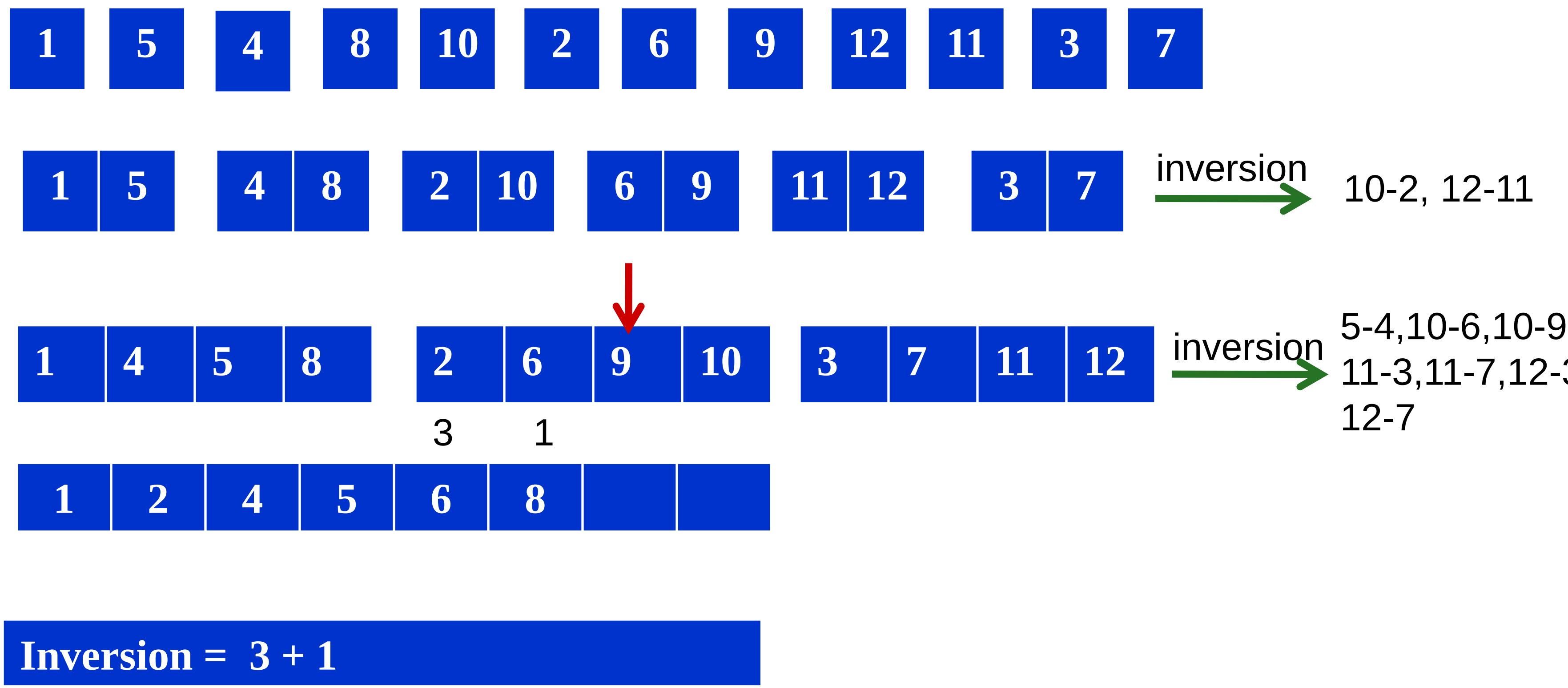


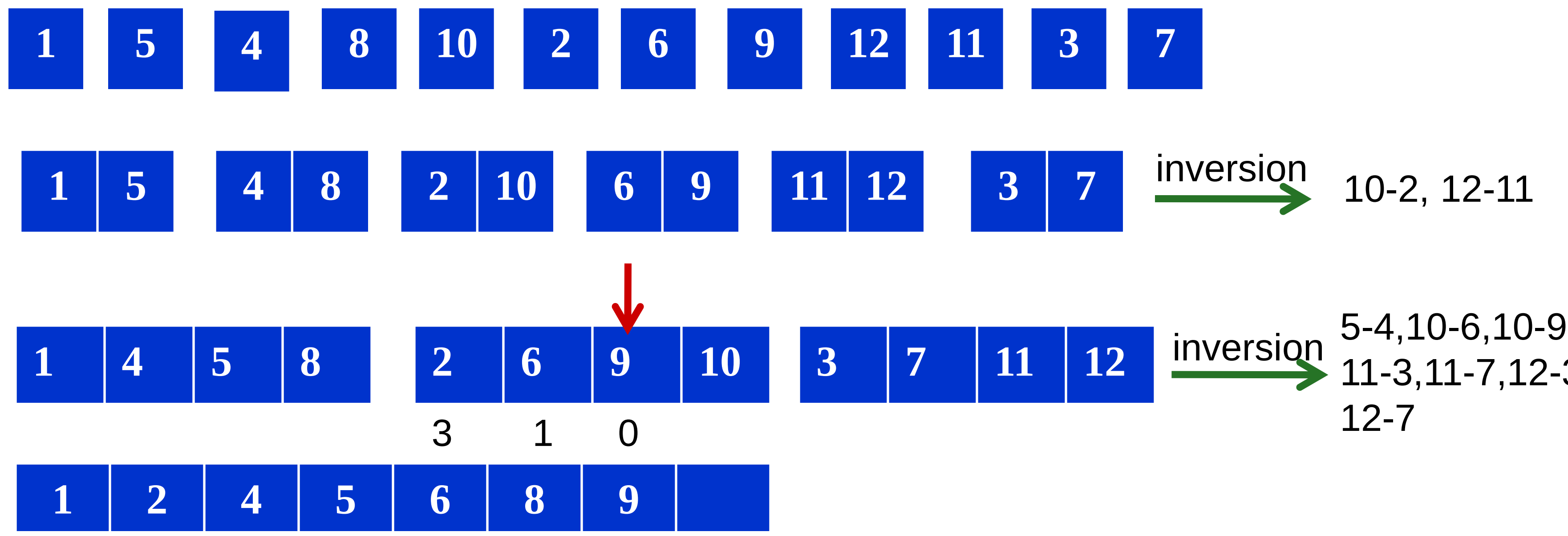
1 5 4 8 10 2 6 9 12 11 3 7
1 5 4 8 2 10 6 9 11 12 3 7
inversion
10-2, 12-11
i=1
1 4 5 8 2 6 9 10 3 7 11 12
inversion
5-4,10-6,10-9
11-3,11-7,12-
12-7
3
1 2 4 5

Inversion = 3 +

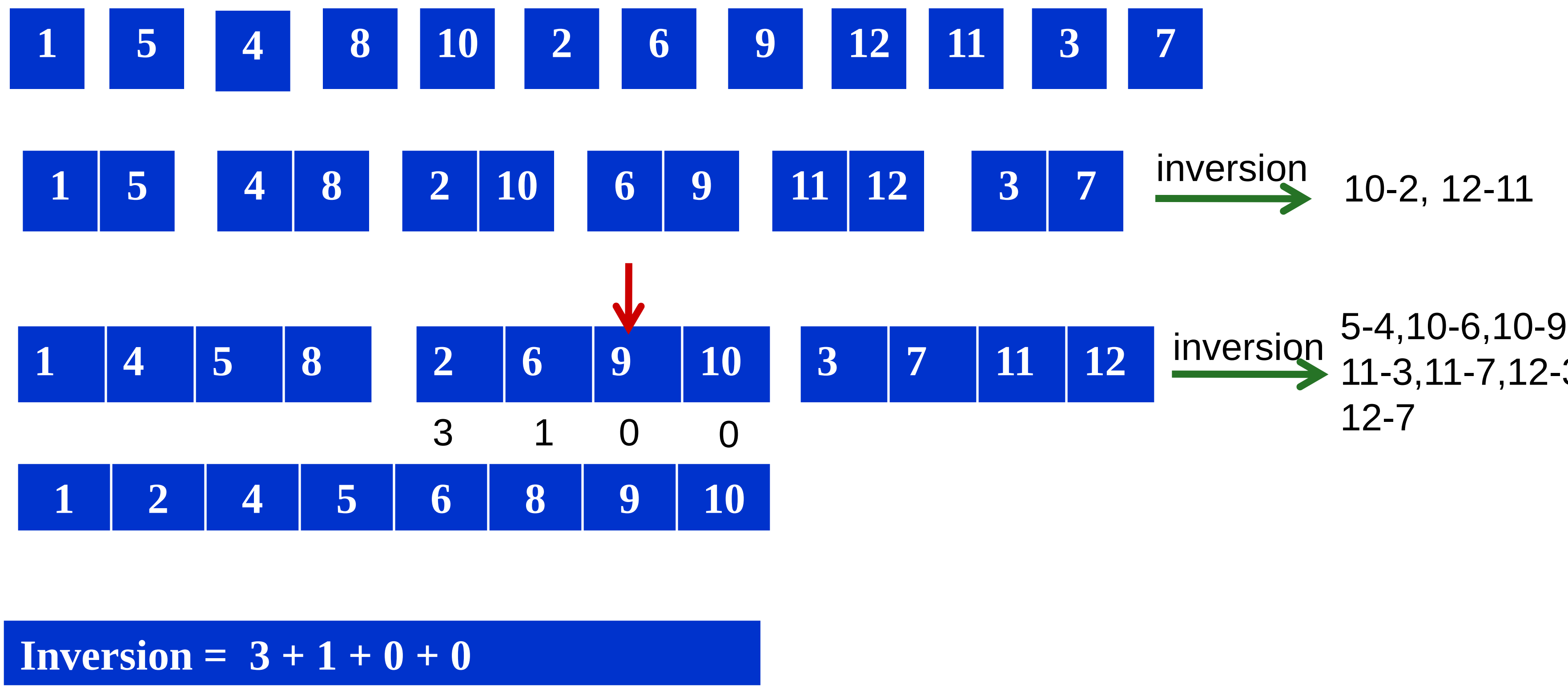


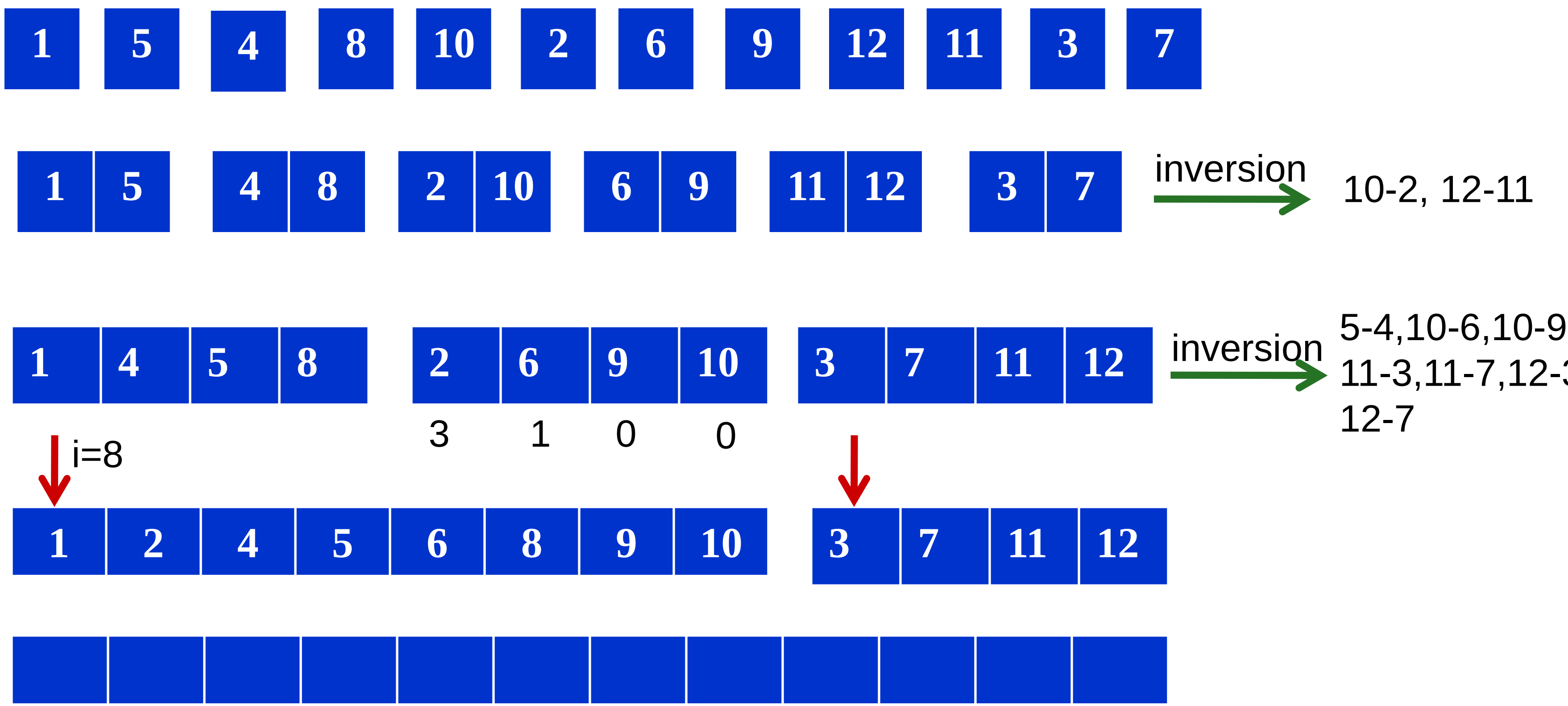
Inversion = 3 + 1


1 5 4 8 10 2 6 9 12 11 3 7
1 5 4 8 2 10 6 9 11 12 3 7
inversion
10-2, 12-11
1 4 5 8 2 6 9 10 3 7 11 12
inversion
5-4,10-6,10-9
11-3,11-7,12-
12-7
3 1
1 2 4 5 6 8
Inversion = 3 + 1



Inversion = 3 + 1 + 0


1 5 4 8 10 2 6 9 12 11 3 7
1 5 4 8 2 10 6 9 11 12 3 7
inversion
10-2, 12-11
1 4 5 8 2 6 9 10 3 7 11 12
inversion
5-4,10-6,10-9,
11-3,11-7,12-3
12-7
3 1 0 0
1 2 4 5 6 8 9 10
Inversion = 3 + 1 + 0 + 0



Inversion = 3 + 1 + 0 + 0

1	5	4	8	10	2	6	9	12	11	3	7

[1 5] [4 8] [2 10] [6 9] [11 12] [3 7]

inversion → 10-2, 12-11

[1 4 5 8] [2 6 9 10] [3 7 11 12]

3 1 0 0

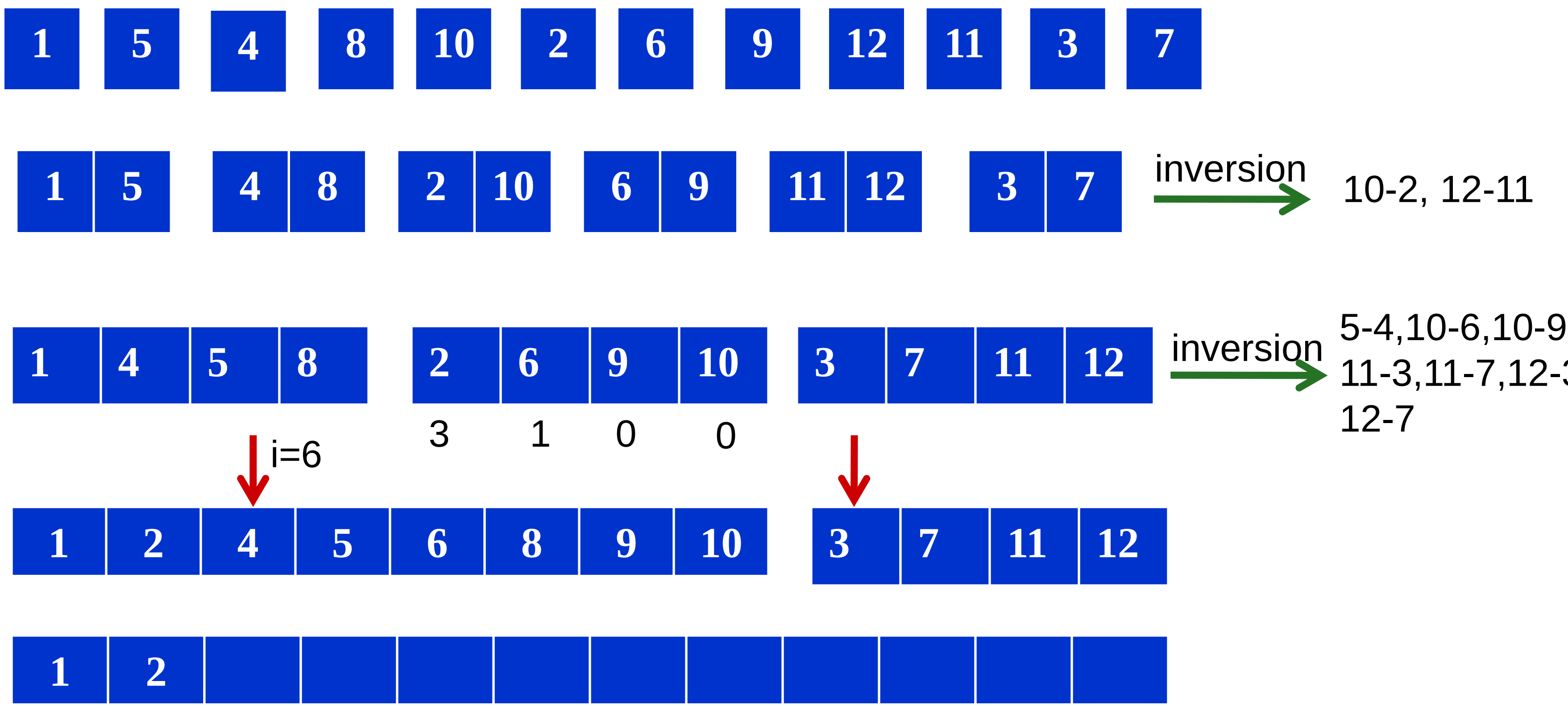
inversion → 5-4,10-6,10-9
11-3,11-7,12-3
12-7

i=7

[1 2 4 5 6 8 9 10] [3 7 11 12]

1											

Inversion = 3 + 1 + 0 + 0



Inversion = 3 + 1 + 0 + 0

1	5	4	8	10	2	6	9	12	11	3	7

1	5		4	8		2	10		6	9		11	12		3	7

inversion → 10-2, 12-11

1	4	5	8		2	6	9	10		3	7	11	12
					3	1	0	0					

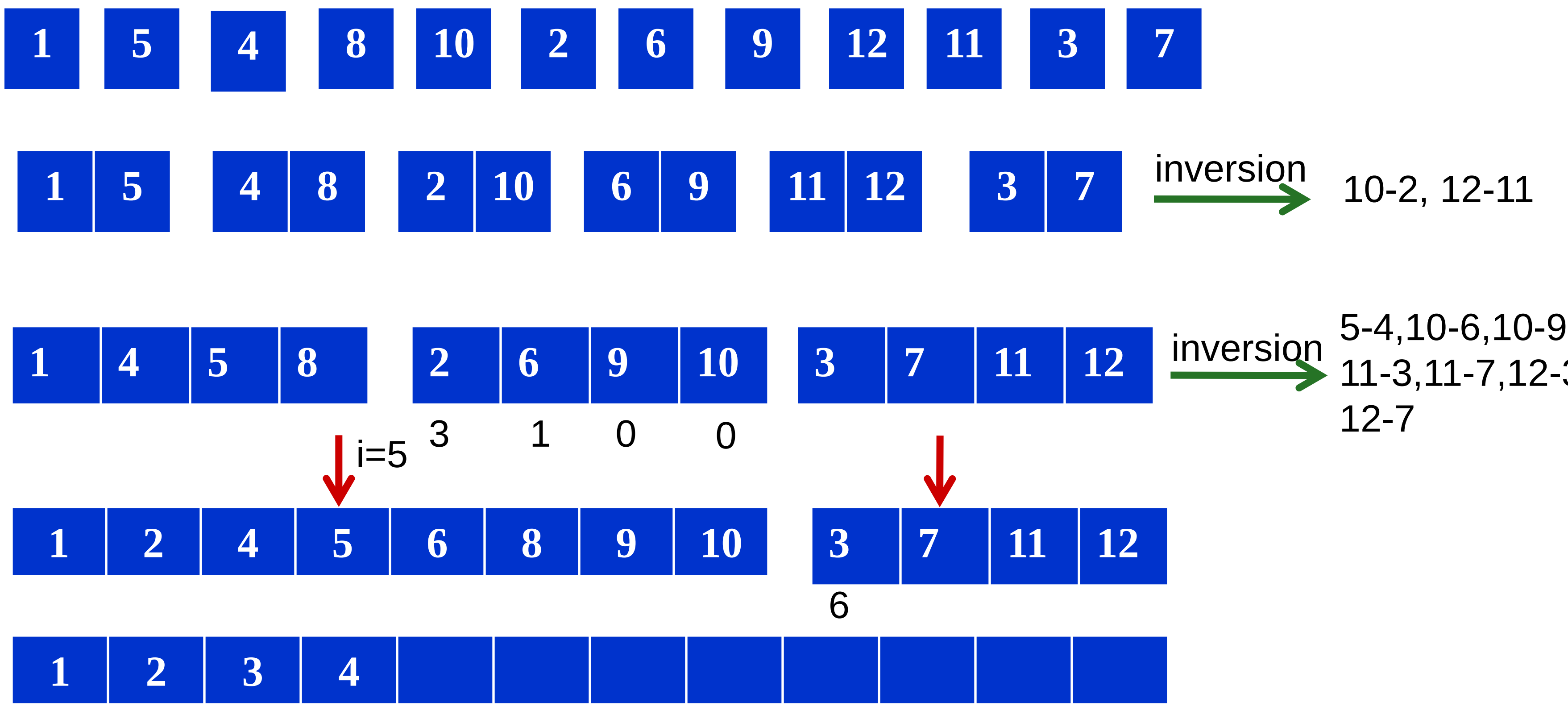
inversion → 5-4,10-6,10-9
11-3,11-7,12-3
12-7

i=6

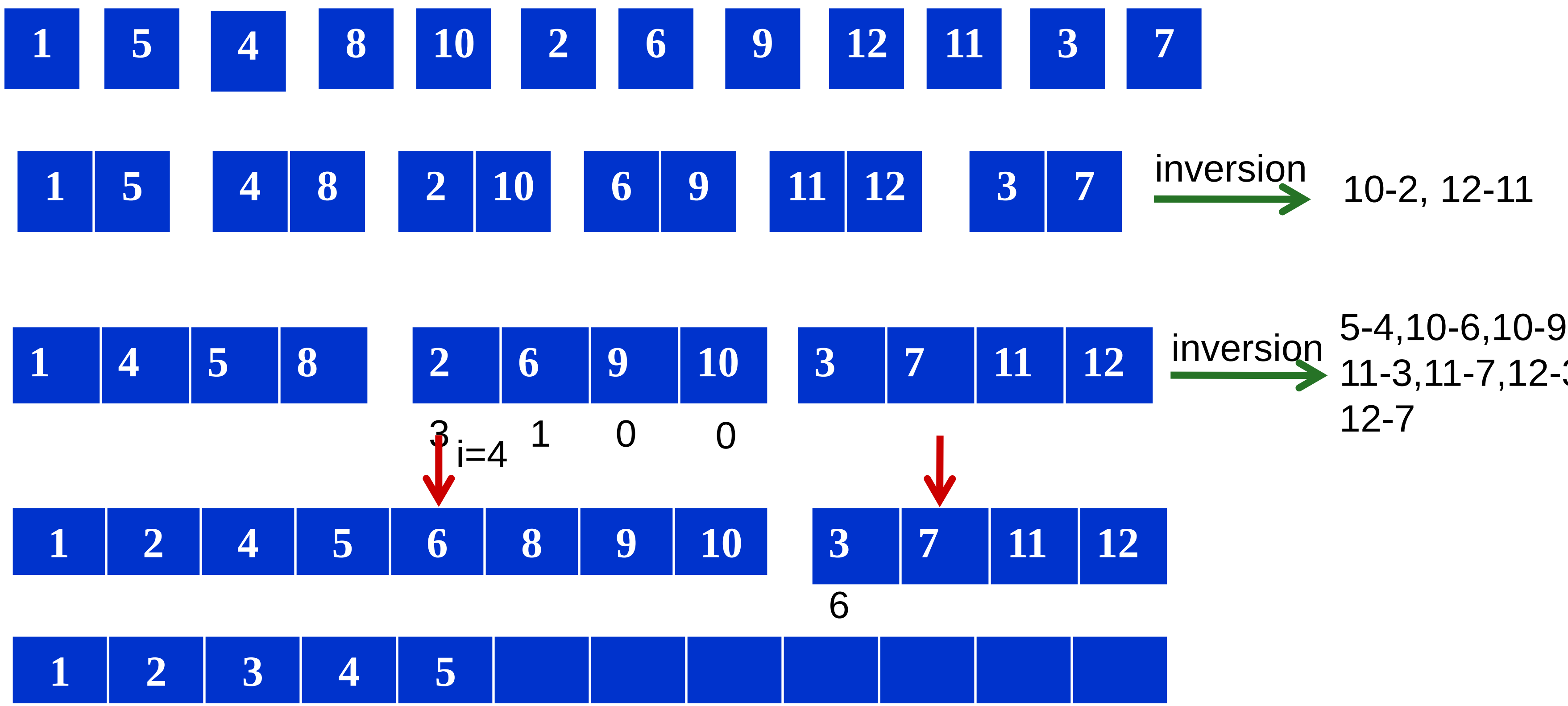
1	2	4	5	6	8	9	10		3	7	11	12
									6			

1	2	3									

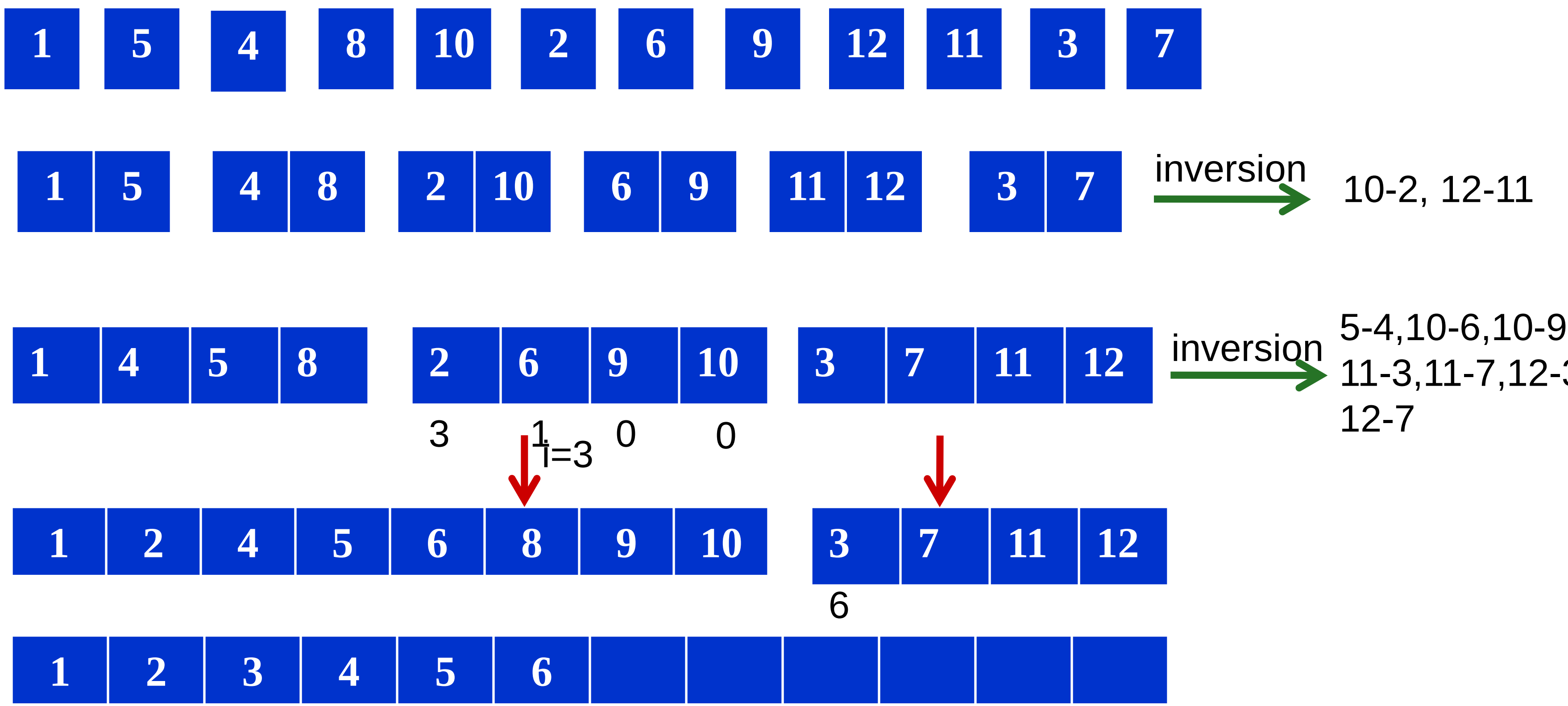
Inversion = 3 + 1 + 0 + 0 + 6



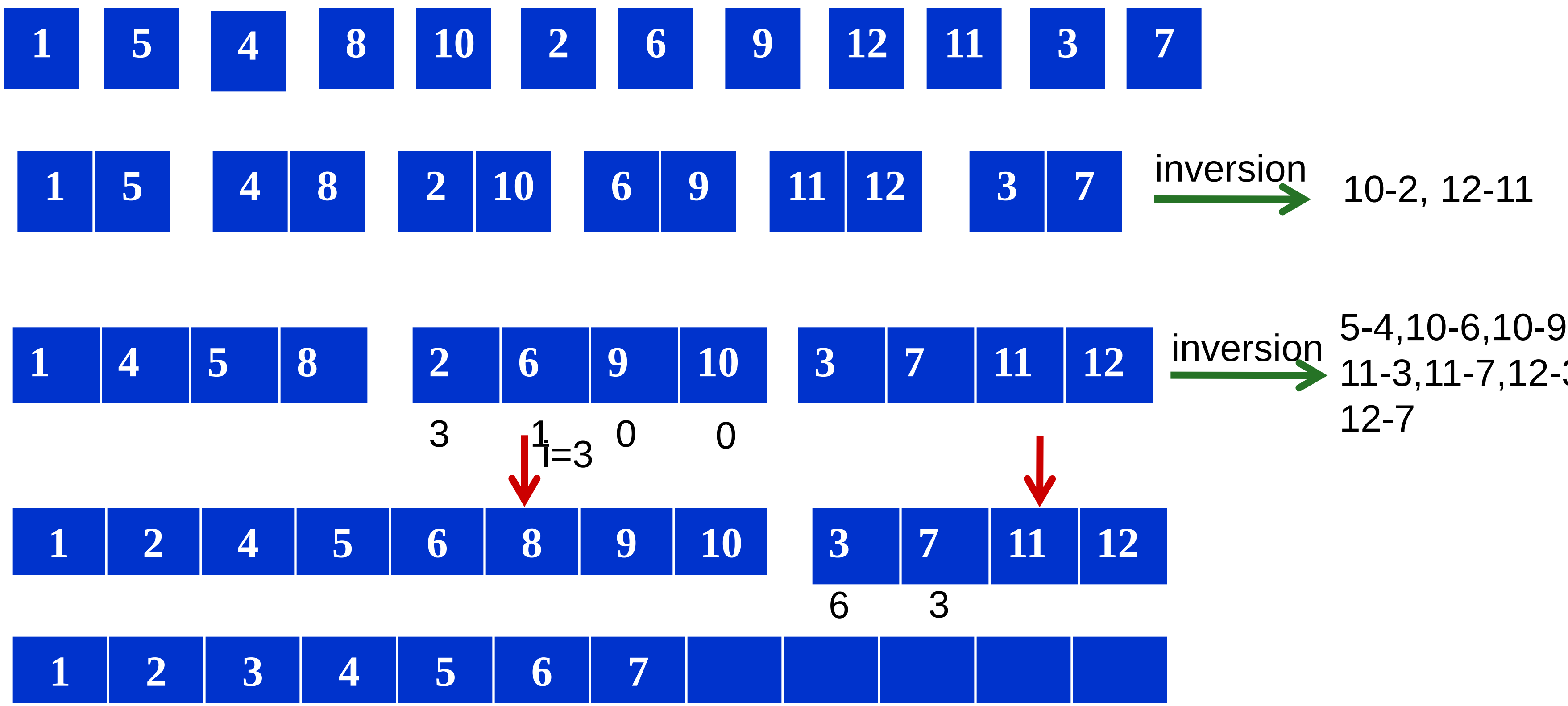
Inversion = 3 + 1 + 0 + 0 + 6



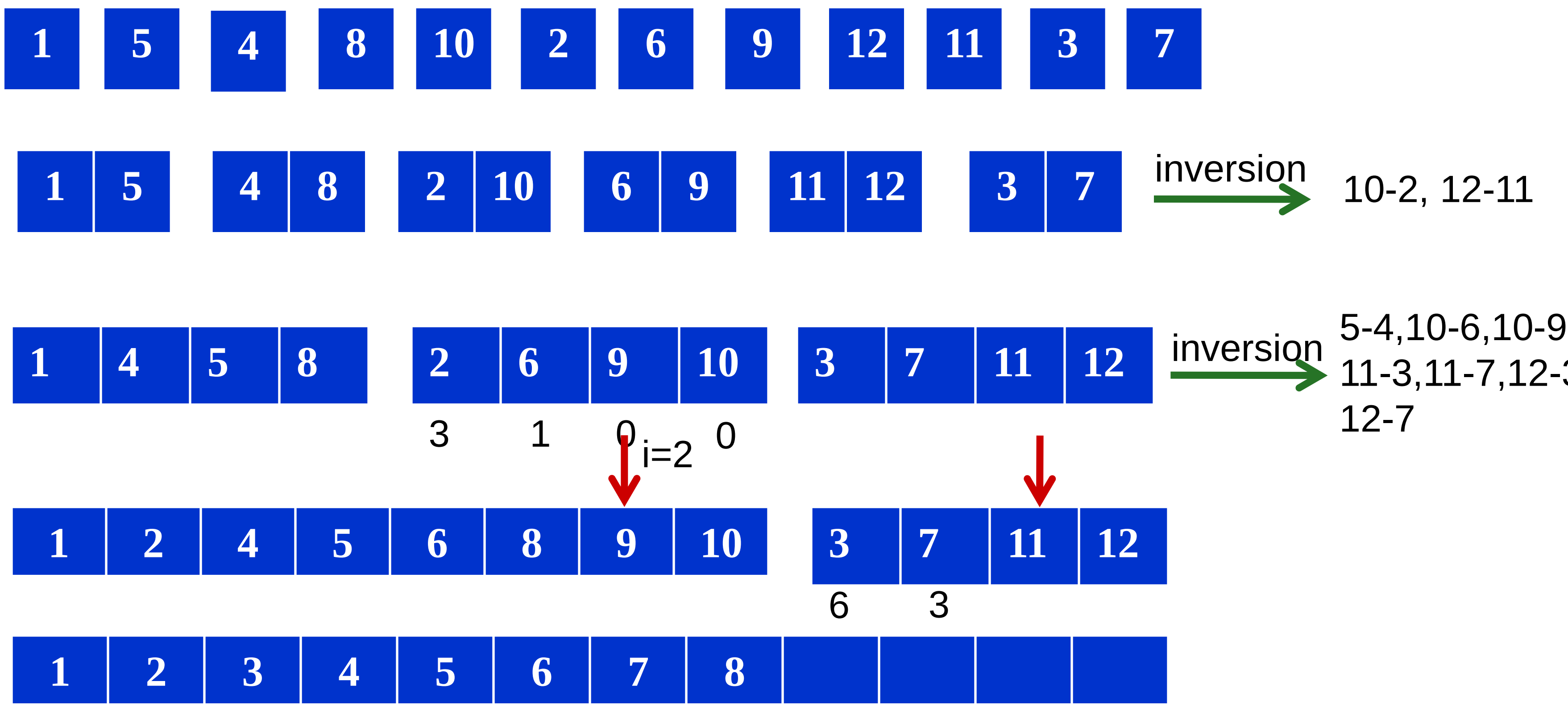
Inversion = 3 + 1 + 0 + 0 + 6



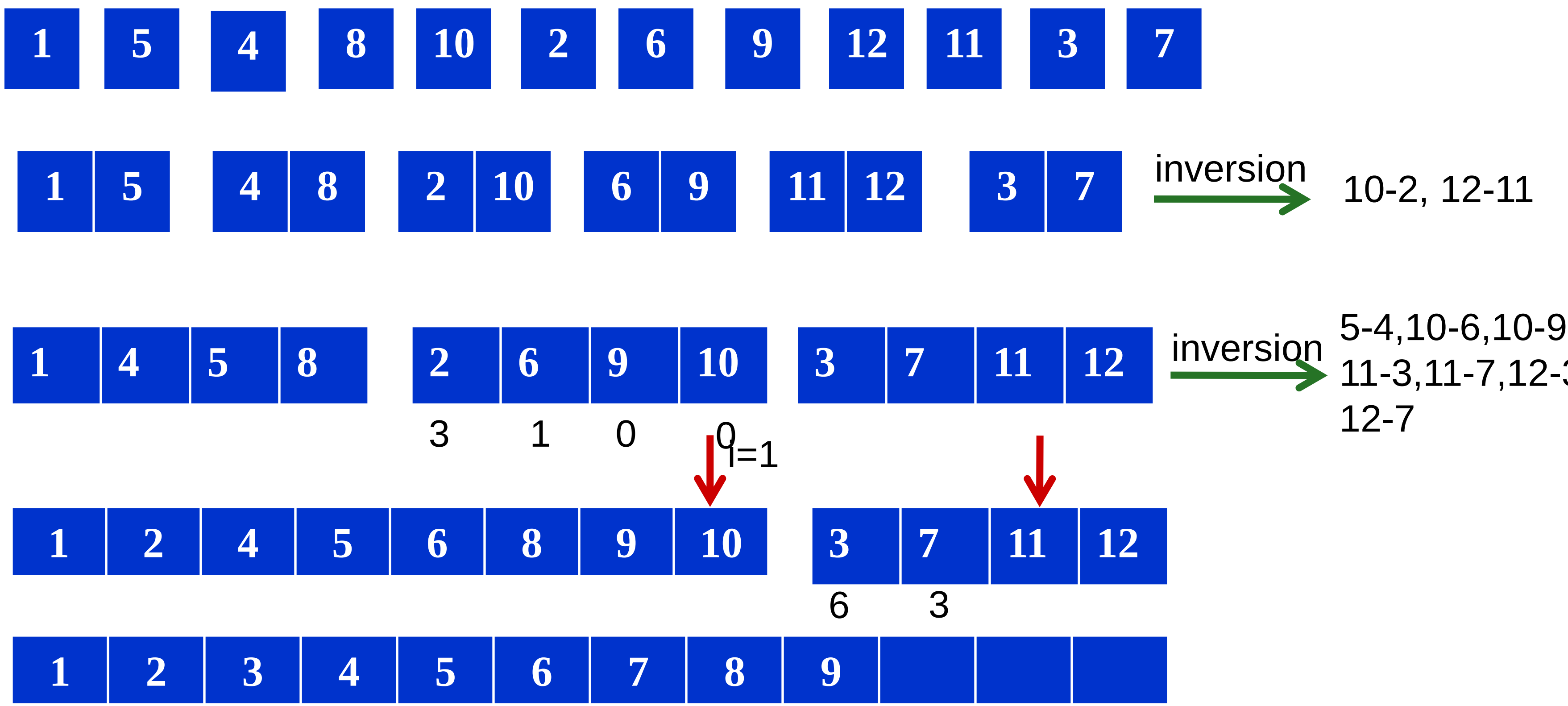
Inversion = 3 + 1 + 0 + 0 + 6



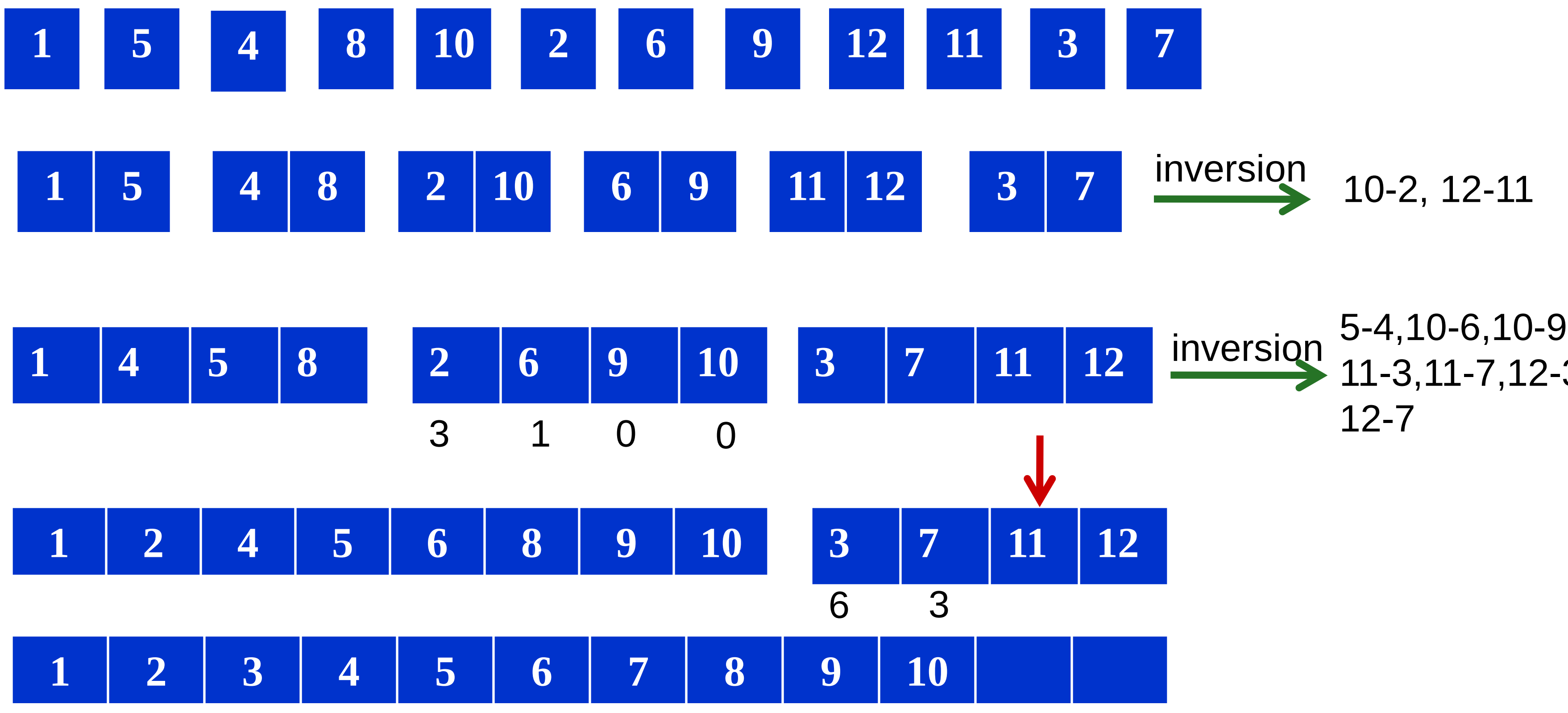
Inversion = 3 + 1 + 0 + 0 + 6 + 3



Inversion = 3 + 1 + 0 + 0 + 6 + 3



Inversion = 3 + 1 + 0 + 0 + 6 + 3



Inversion = 3 + 1 + 0 + 0 + 6 + 3

1 5 4 8 10 2 6 9 12 11 3 7

1 5 | 4 8 | 2 10 | 6 9 | 11 12 | 3 7

inversion → 10-2, 12-11

1 4 5 8 | 2 6 9 10 | 3 7 11 12

3 1 0 0

inversion → 5-4,10-6,10-9
11-3,11-7,12-3
12-7

1 2 4 5 6 8 9 10 | 3 7 11 12

6 3 0

1 2 3 4 5 6 7 8 9 10 11

Inversion = 3 + 1 + 0 + 0 + 6 + 3 + 0

1 5 4 8 10 2 6 9 12 11 3 7

1 5 | 4 8 | 2 10 | 6 9 | 11 12 | 3 7

inversion → 10-2, 12-11

1 4 5 8 | 2 6 9 10 | 3 7 11 12

3 1 0 0

inversion → 5-4,10-6,10-9
11-3,11-7,12-
12-7

1 2 4 5 6 8 9 10 | 3 7 11 12

6 3 0 0

1 2 3 4 5 6 7 8 9 10 11 12

Inversion = 3 + 1 + 0 + 0 + 6 + 3 + 0 + 0

1	5	4	8	10	2	6	9	12	11	3	7

1	5	4	8	2	10	6	9	11	12	3	7

inversion → 10-2, 12-11

1	4	5	8	2	6	9	10	3	7	11	12
				3	1	0	0				

inversion → 5-4,10-6,10-9
11-3,11-7,12-
12-7

1	2	4	5	6	8	9	10	3	7	11	12
								6	3	0	0

1	2	3	4	5	6	7	8	9	10	11	12

Inversion = 3 + 1 + 0 + 0 + 6 + 3 + 0 + 0

Total Inversion = 13 + 7 + 2 = 22

Counting Inversion

- Merge-and-Count (A,B)
  - Maintain a *Current pointer into each list*, *initialized to* point to the front elements
  - Maintain a variable *Count for the number of inversions, initialized* to 0
  - While both lists are nonempty {
    - Let a_i *and* b_j *be the elements pointed to by the Current pointer*
    - Append the smaller of these two to the output list
    - If $b_j < a_i$ then
      - Increment *Count by the number of elements remaining in A*
    - Endif
  - Advance the *Current pointer in the list from which the*
  - smaller element was selected.}
  - once one list is empty, append the remainder of the other list to the output
  - Return *Count and the merged list*

Counting Inversion

- **Sort-and-Count(L) {**
  - **if list L has one element return 0 and the list L**
  - **Divide the list into two halves A and B**
  - **(r_A, A) = Sort-and-Count(A)**
  - **(r_B, B)= Sort-and-Count(B)**
  - **(r, L) = Merge-and-Count(A, B)**
  - **return $r = r_A + r_B + r$ and the sorted list L**

  }
- **The Recurrence relation**

$$T(n) \le T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n) \quad \Rightarrow \mathrm{T}(n) = O(n \log n)$$