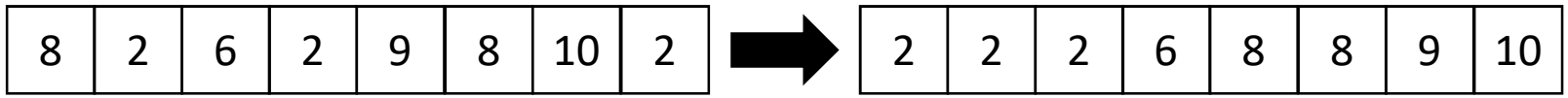
Sorting Multisets

CSCI2100 Tutorial 5

Problem: Sorting Multiset

- Problem Input:
  - An array containing n integers from a total order
- Goal:
  - An array containing all the integers in nondescending order

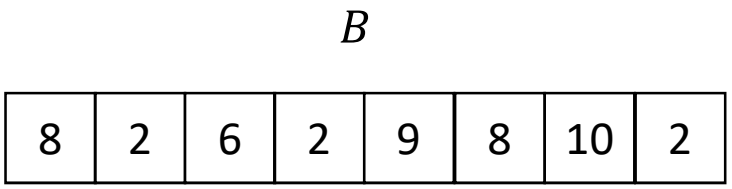


Recall: Merge Sort

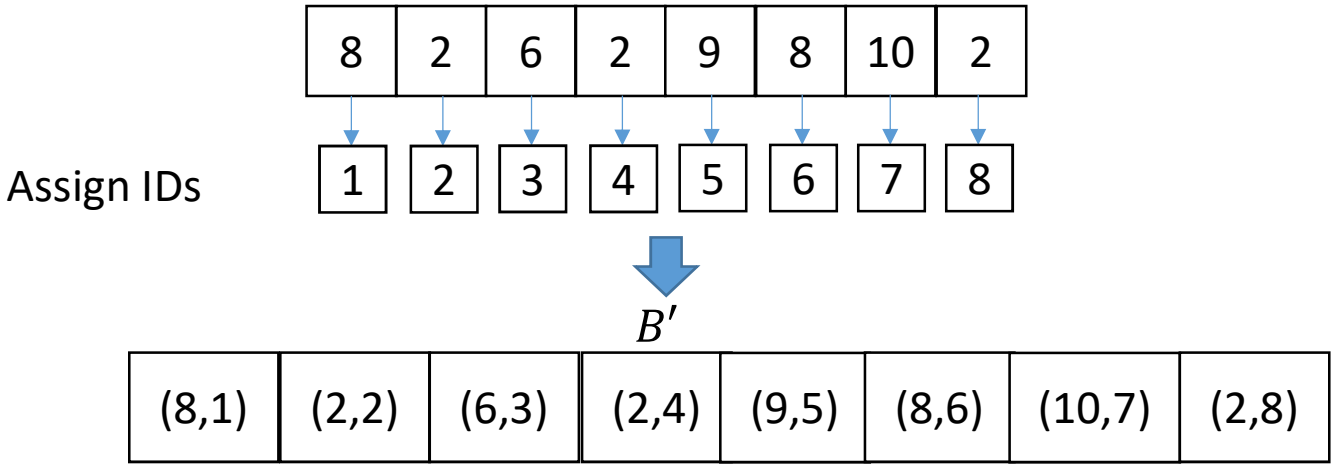
- Merge sort works as long as the objects are
  - distinct, and
  - can be compared.

- **The algorithm does not care what the objects actually are (integers, documents, images, etc.).**

- Same is true for any comparison-based algorithms, e.g., selection sort, distribution sort, quick sort, etc.



- Cannot use merge sort on B as the objects are not distinct.
- **Idea:** convert them into distinct objects.



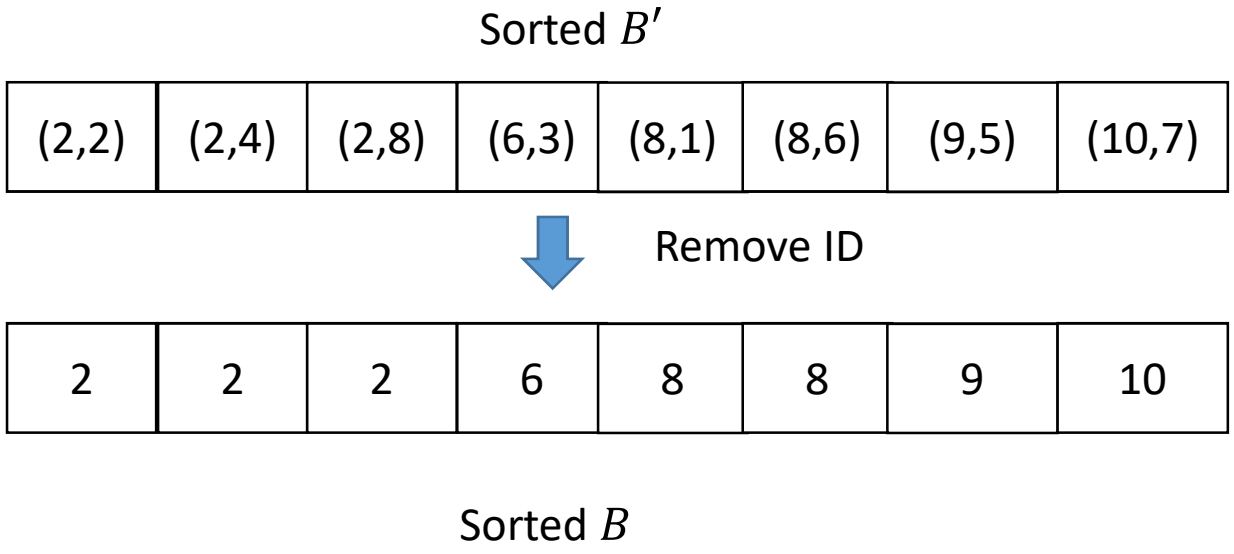
- Still need to define how to compare the new objects.
  - Want to make sure the sorter order on the new objects is a legal sorted order on the original objects.

Define a Comparator

- Given two objects $e_1 = (v_1, id_1)$ and $e_2 = (v_2, id_2)$:
- If $v_1 < v_2$, then rule $e_1 < e_2$
- If $v_1 > v_2$, then rule $e_1 > e_2$
- If $int_1 = int_2$:
  - If $id_1 < id_2$, then rule $e_1 < e_2$
  - If $id_1 > id_2$, then rule $e_1 > e_2$

E.g.: (2,1) < (2,10) and (1, 10) < (2, 1)

Apply Merge Sort on B’



Time Complexity

- Modifying the input array takes $O(n)$
- Merging sort takes $O(n \log n)$ time.
- Removing the IDs to produce the final output takes $O(n)$ time.

- Overall time complexity: $O(\mathrm{n} \log n)$

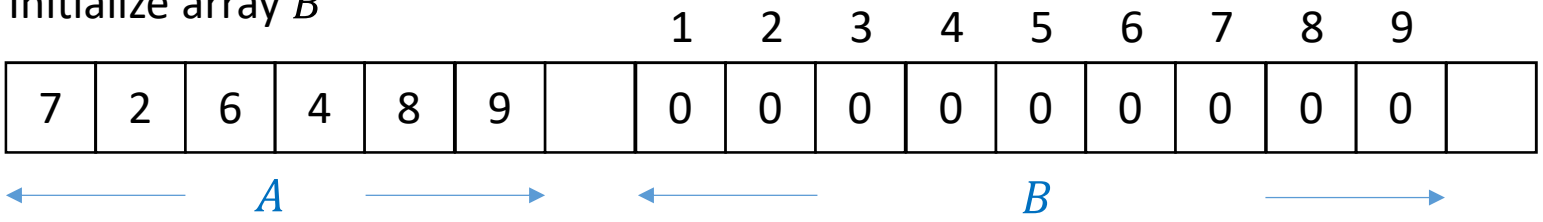
Sorting a Multiset of integers in small range

- Problem Input:
  - An array containing a multiset of n integers, each from the domain $[1, U]$.
- Goal:
  - An array containing all the integers in nondescending order.

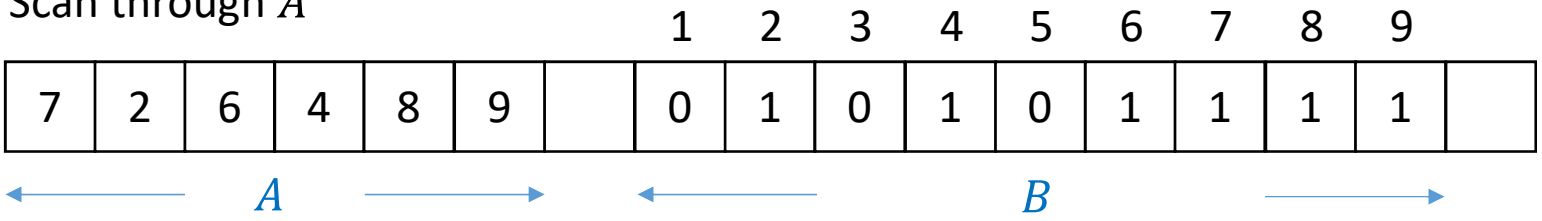
Review: the Counting Sort in the Lecture

- Sort a set of integers in a domain $[1, 9]$

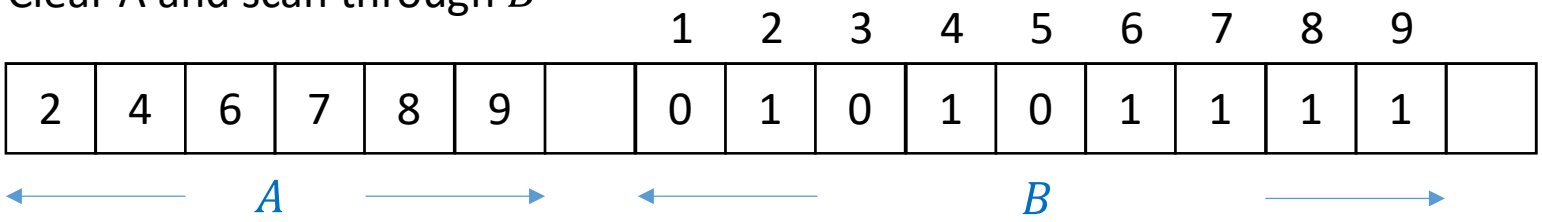
Initialize array B



Scan through A



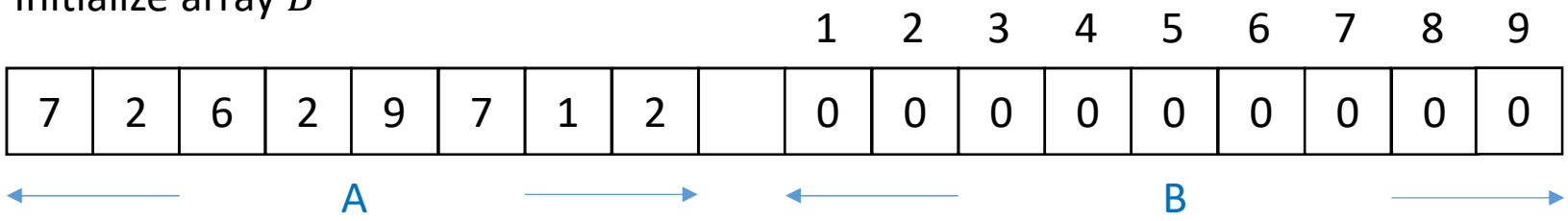
Clear A and scan through B



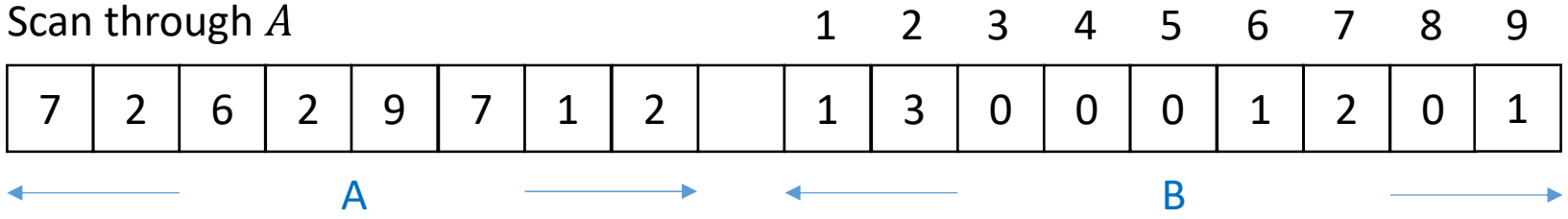
A Simple Extension

- Sort a multiset of integers in domain [1, 9]
- B stores counters.

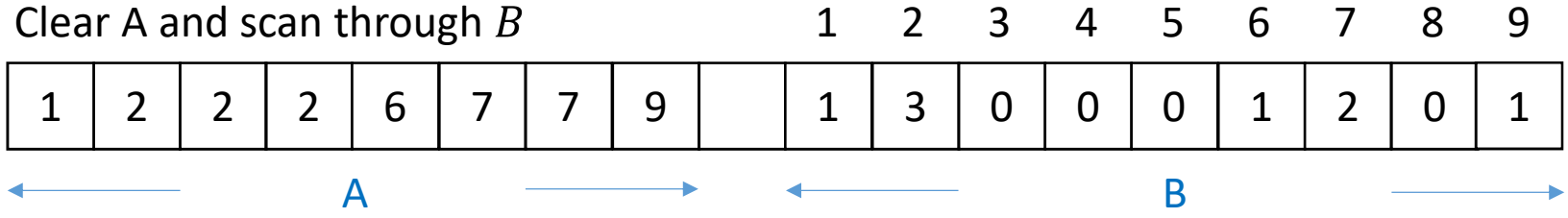
Initialize array B



Scan through A



Clear A and scan through B



The “Real” Multi-set Sorting Problem

- Problem Input:
  - An array containing n key-value pairs, where each key is an integer from [1, U].

    E.g.: (93, 1155123456)

- Goal:
  - An array storing all pairs in nondescending order of key.

Example

- Input:
  $\{(9, v_1), (7, v_2), \{2, v_3\}, \{6, v_4\}, \{2, v_5\}, \{7, v_6\}, \{1, v_7\}, \{2, v_8\}\}$
- Initially we will have the following array

Input Array

k_1	v_1	k_2	v_2	k_3	v_3	k_4	v_4	k_5	v_5	k_6	v_6	k_7	v_7	k_8	v_8
9	v_1	7	v_2	2	v_3	6	v_4	2	v_5	7	v_6	1	v_7	2	v_8

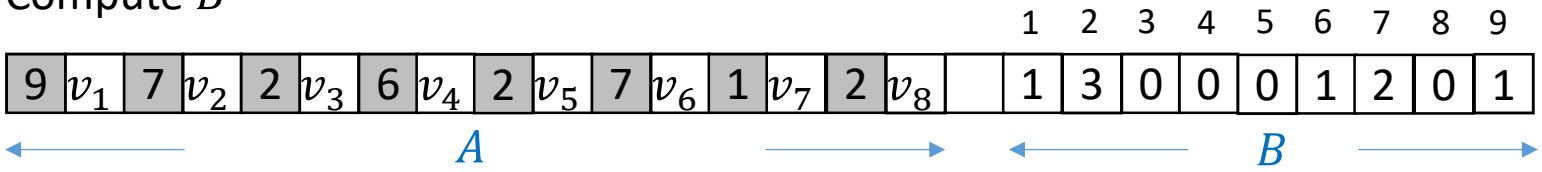
- Rearrange the elements so that their keys are sorted:

Sorted Array

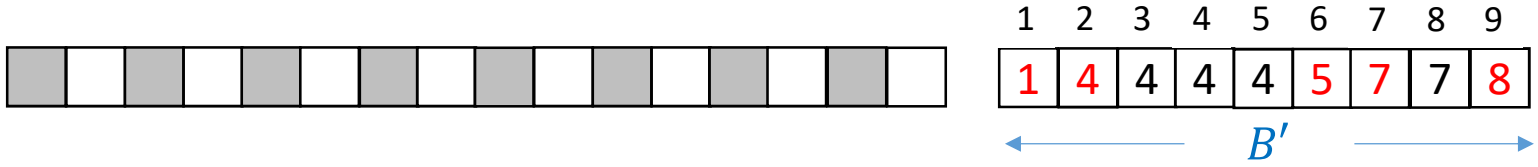
1	v_7	2	v_3	2	v_5	2	v_8	6	v_4	7	v_2	7	v_6	9	v_1

Example

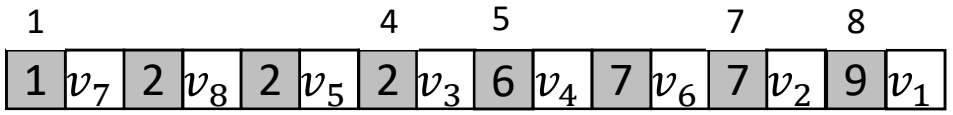
Compute B



Compute the prefix sum of B



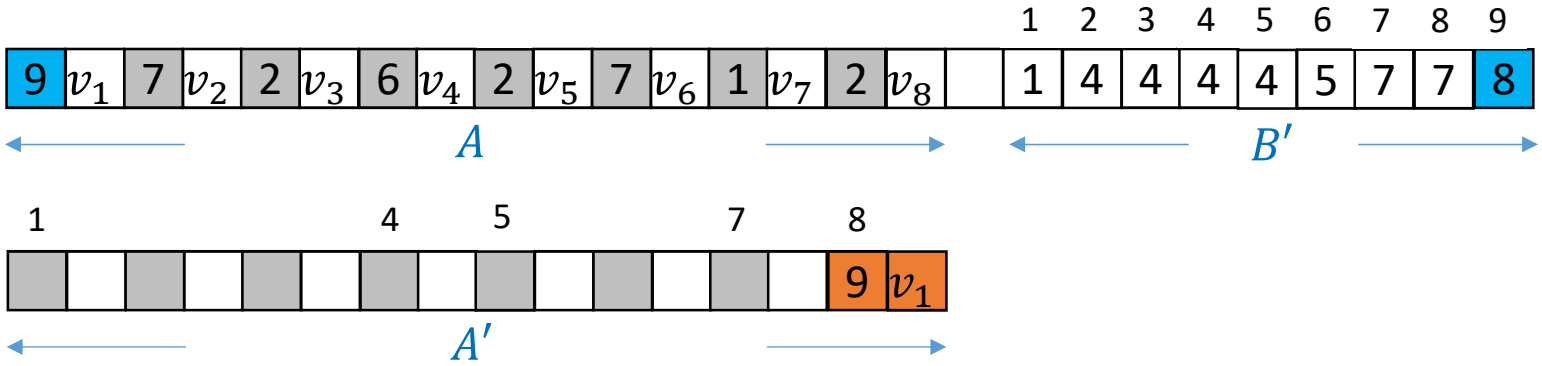
$B'[i]$ indicates the last index of a particular key in the final sorted array.



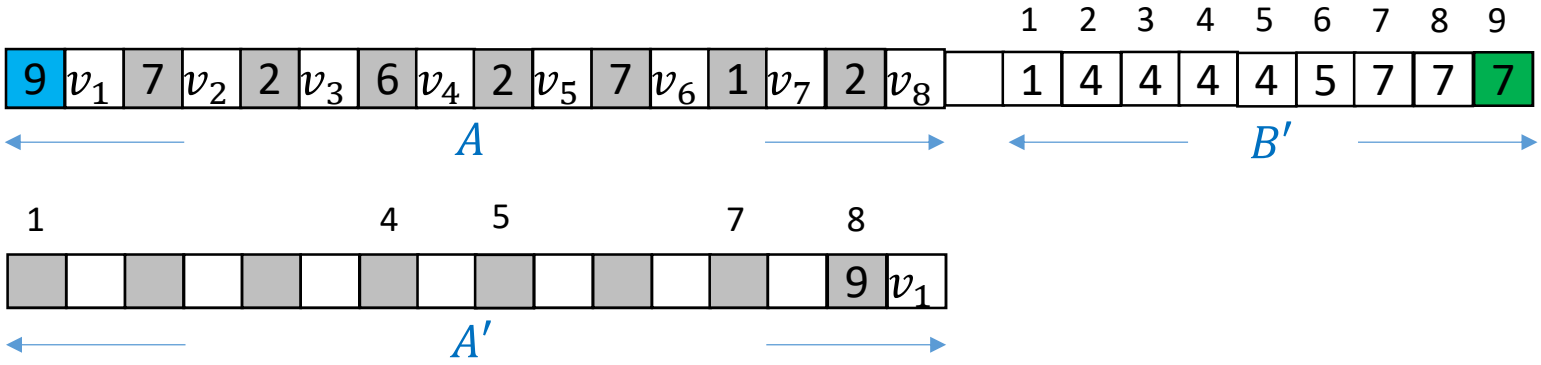
The final sorted array

Example

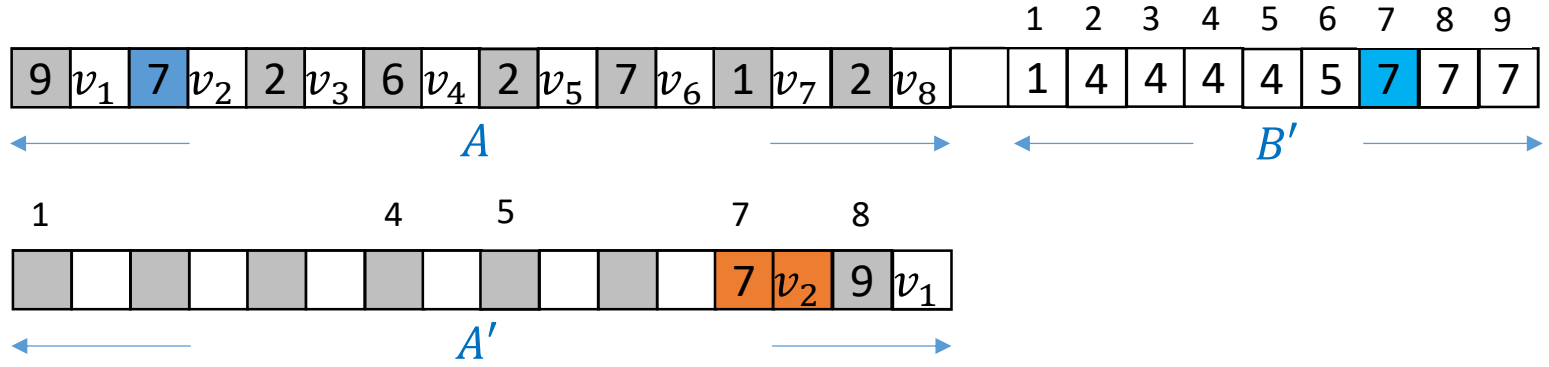
Build array A' by repeating: for a key-value pair (k, v) in A, copy it to the $B'[k]$-th position in A'

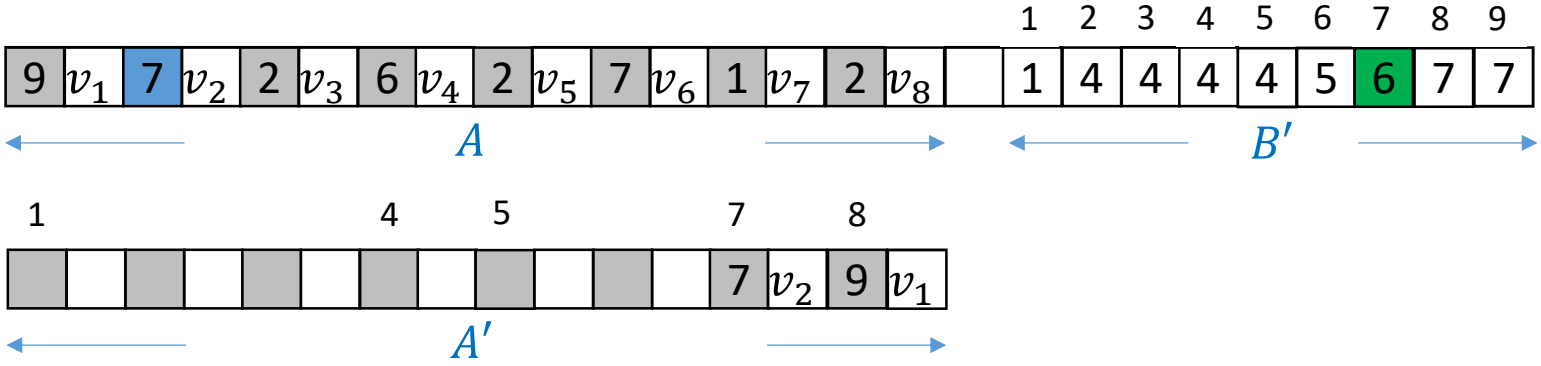


Decrement $B'[k]$ such that it points the position for the next pair with key k.

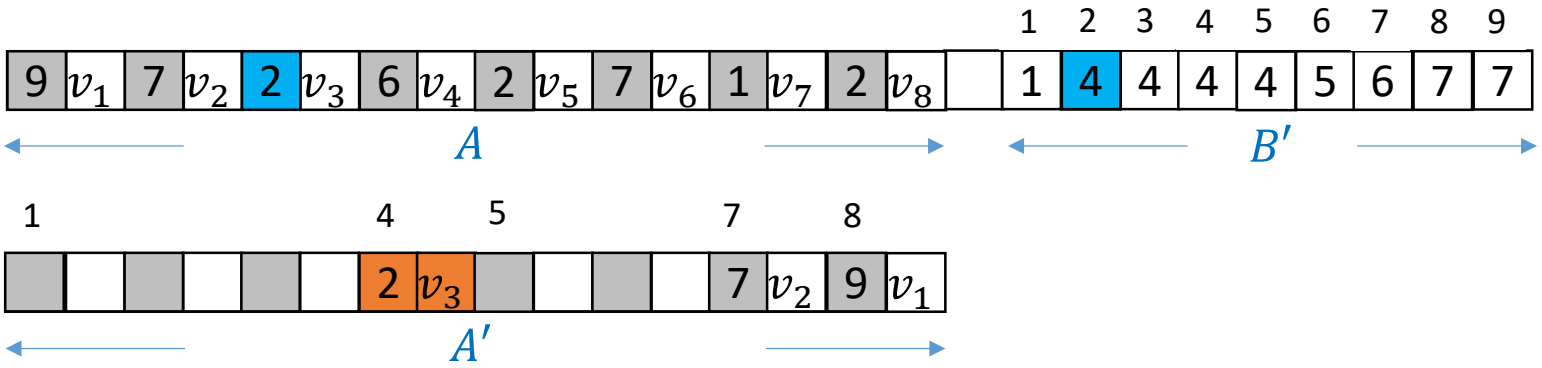


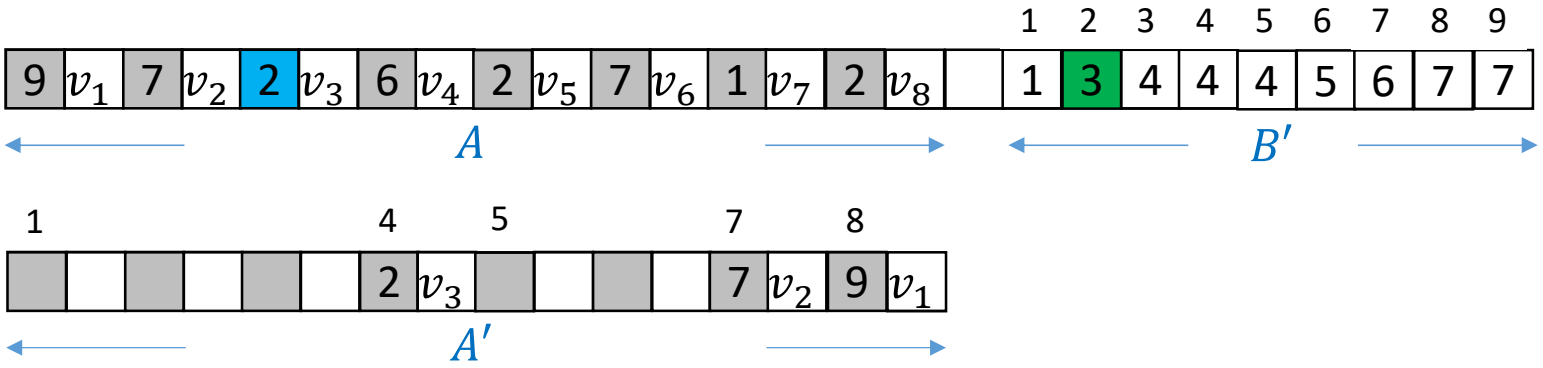
Example


1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 4 4 4 4 5 7 7 7
A
B'
1 4 5 7 8
7 v_2 9 v_1
A'

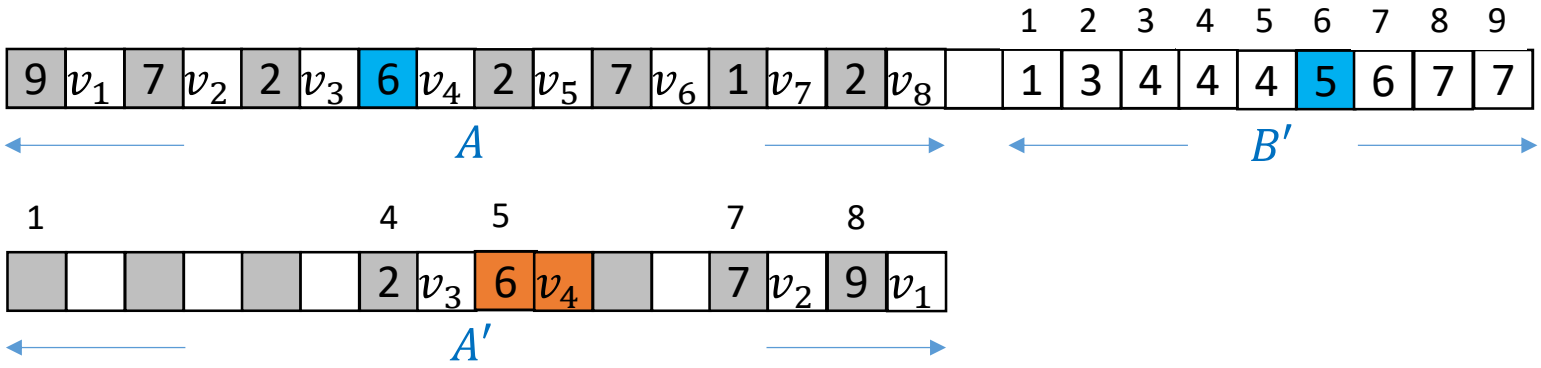

1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 4 4 4 4 5 6 7 7
A
B'
1 4 5 7 8
7 v_2 9 v_1
A'

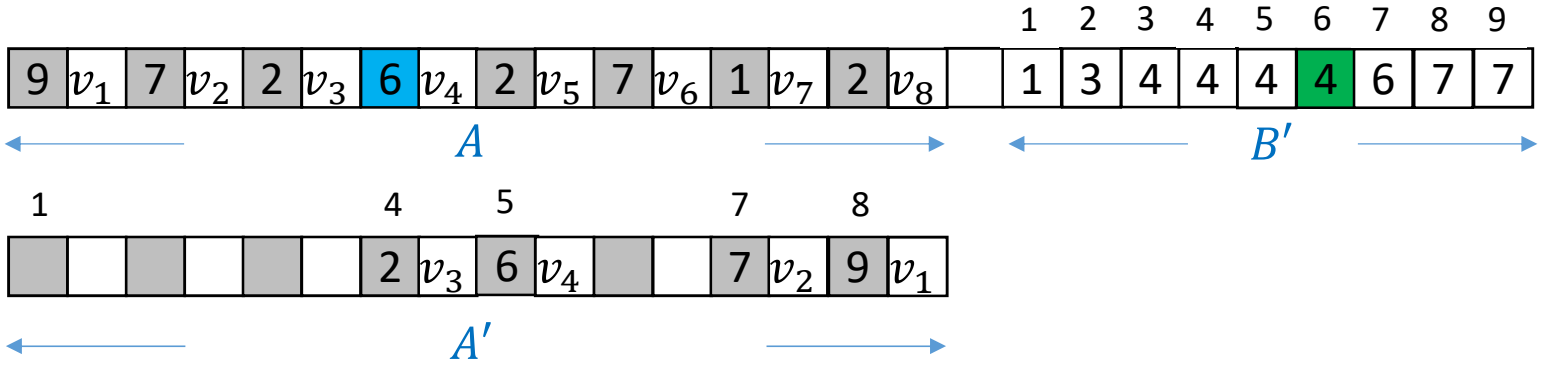
Example


1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 4 4 4 4 5 6 7 7
A
B'
1 4 5 7 8
2 v_3 7 v_2 9 v_1
A'

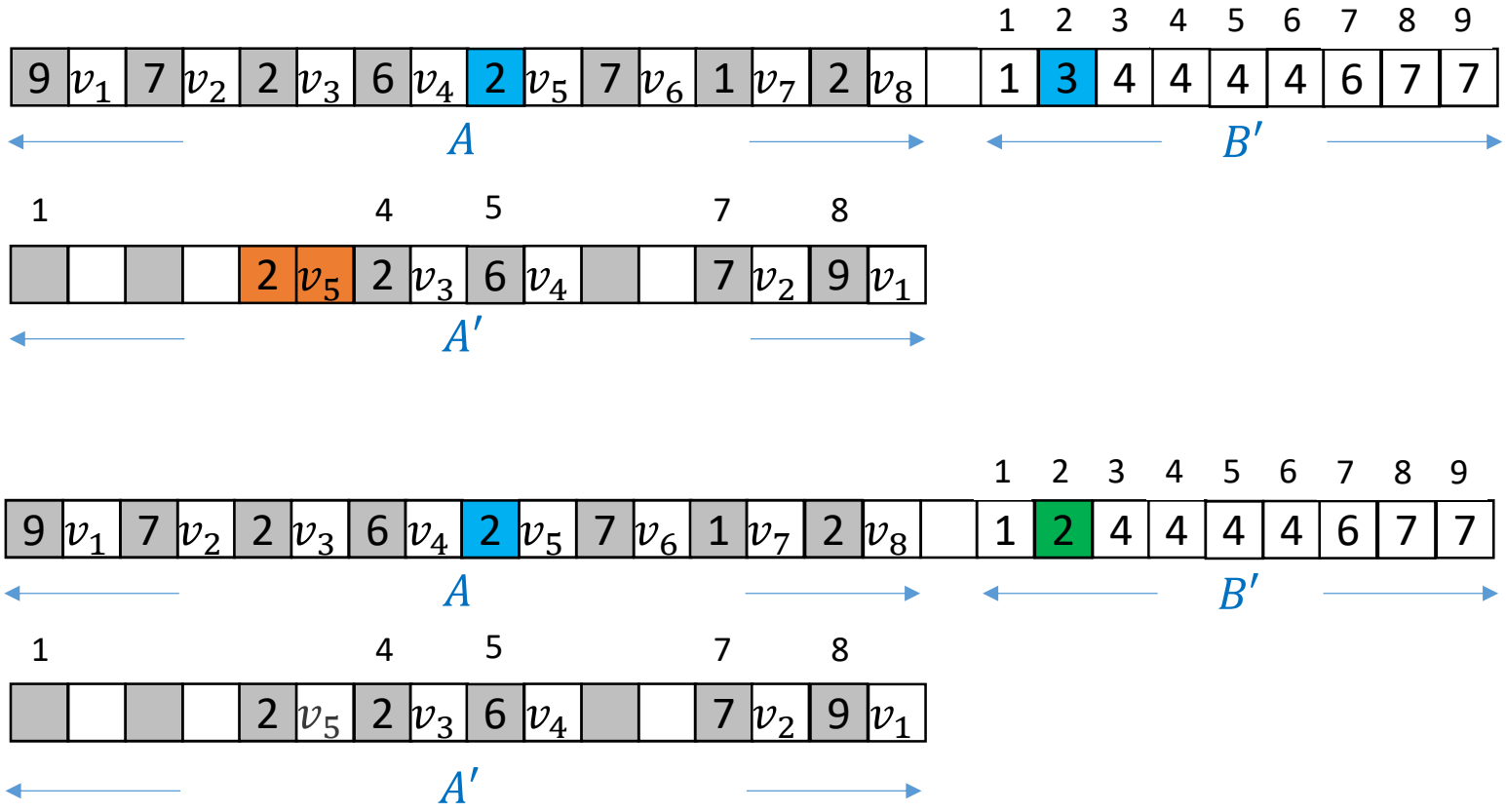

1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 3 4 4 4 5 6 7 7
A
B'
1 4 5 7 8
2 v_3 7 v_2 9 v_1
A'

Example

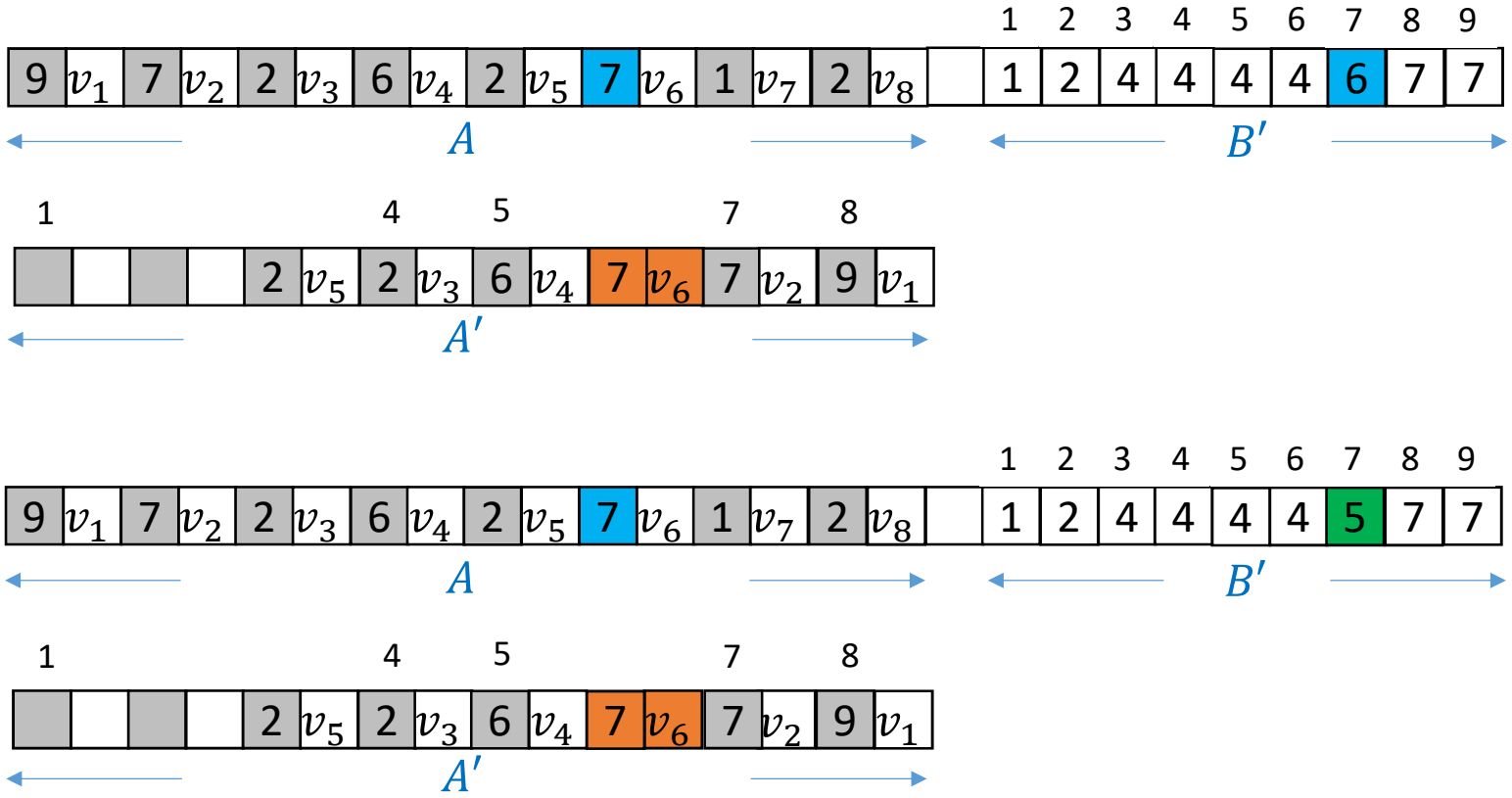

1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 3 4 4 4 5 6 7 7
A
B'
1 4 5 7 8
2 v_3 6 v_4 7 v_2 9 v_1
A'


1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 3 4 4 4 4 6 7 7
A
B'
1 4 5 7 8
2 v_3 6 v_4 7 v_2 9 v_1
A'

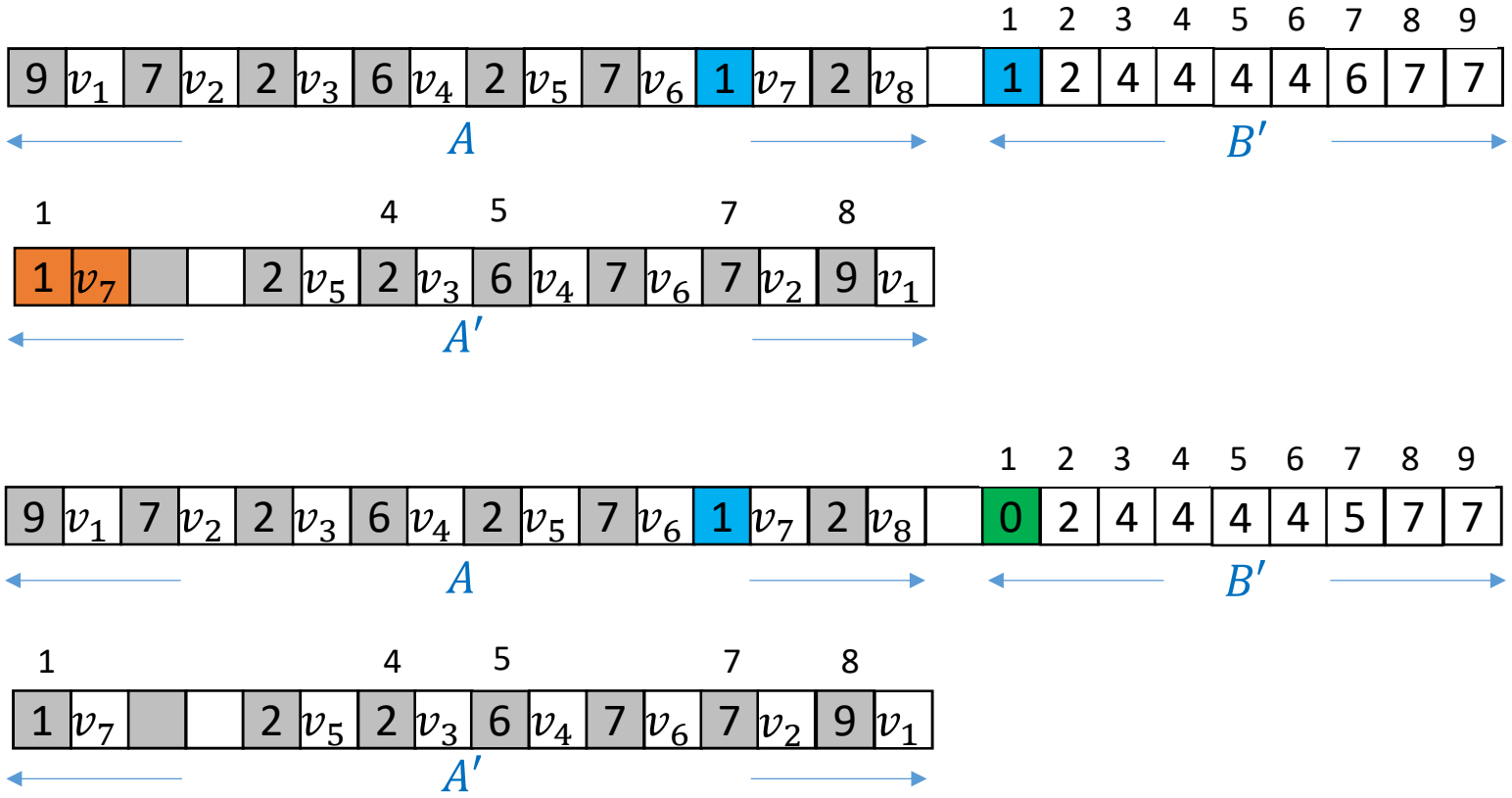
Example


1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 3 4 4 4 4 6 7 7
A
B'
1 4 5 7 8
2 v_5 2 v_3 6 v_4 7 v_2 9 v_1
A'
1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 2 4 4 4 4 6 7 7
A
B'
1 4 5 7 8
2 v_5 2 v_3 6 v_4 7 v_2 9 v_1
A'

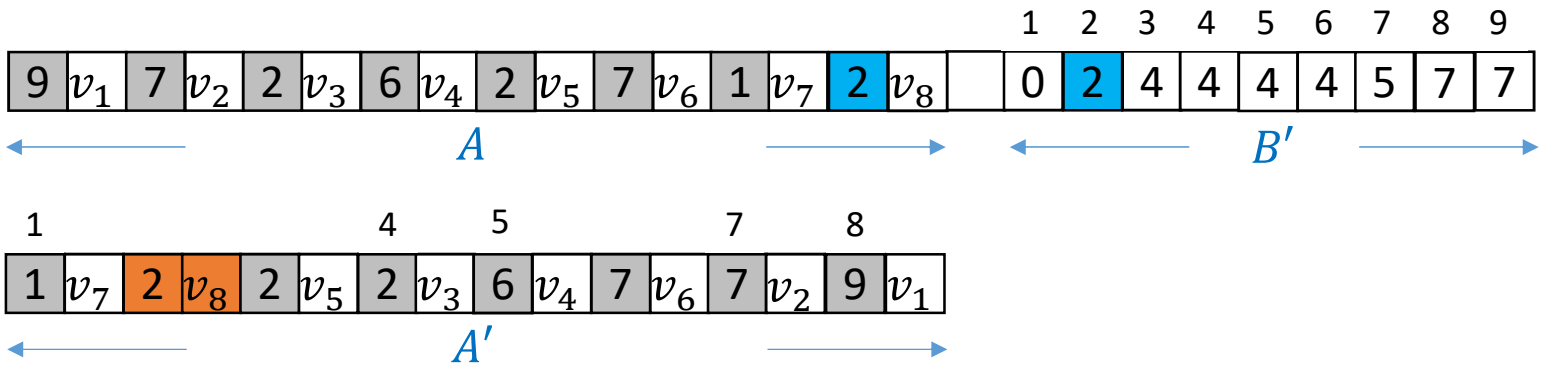
Example


1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 2 4 4 4 4 6 7 7
A
B'
1 4 5 7 8
2 v_5 2 v_3 6 v_4 7 v_6 7 v_2 9 v_1
A'
1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 2 4 4 4 4 5 7 7
A
B'
1 4 5 7 8
2 v_5 2 v_3 6 v_4 7 v_6 7 v_2 9 v_1
A'

Example


1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
1 2 4 4 4 4 6 7 7
A
B'
1 4 5 7 8
1 v_7 2 v_5 2 v_3 6 v_4 7 v_6 7 v_2 9 v_1
A'
1 2 3 4 5 6 7 8 9
9 v_1 7 v_2 2 v_3 6 v_4 2 v_5 7 v_6 1 v_7 2 v_8
0 2 4 4 4 4 5 7 7
A
B'
1 4 5 7 8
1 v_7 2 v_5 2 v_3 6 v_4 7 v_6 7 v_2 9 v_1
A'

Example



Overall time complexity: $O(n + U)$