Трансформираме задачата по следния начин: за да отговорим на въпроса колко е броят на различните числа в интервала [l, r], всъщност трябва да отговорим на въпроса колко са различните числа, срещащи се в интервала [1, r], такива че последното им срещане попада в интервала [l, r]. Нека за елемент a[i] в last[a[i]] пазим позицията на последното му срещане до момента (т.е. този масив ще се променя в процеса на работа). Решението на задачата ще се състои в това чрез подходящо построено индексно дърво да отговаряме бързо на тези заявки. Ще ги обработваме offline, т.е. отговорът няма да се дава веднага след подаване на заявката. Алгоритъмът е следният:

1. Сортираме заявките по техния r.
2. Обхождаме ги в сортирания ред.
3. Нека в момента разглеждаме заявка [li, ri]. На текущата стъпка обработваме елементите в интервала [ri-1+1, ri] – тези, с които се различава крайната част на интервала спрямо предния (ако има такива). За всеки от тях (нека го наречем j):
   1. Ако до момента сме го срещнали поне веднъж, т.е. last[a[j]] != 0, в индексното дърво добавяме -1 на позиция j и го update-ваме. Това го правим, тъй като мястото на последното му срещане се променя, затова първо трябва да премахнем старото.
   2. Променяме позицията на последното му срещане, т.е. last[a[j]] = j.
   3. Добавяме 1 в индексното дърво на позиция j и го update-ваме.
4. Отговорът за текущо-разглежданата заявка е сумата на индексното дърво в интервала [li, ri].

Трябва да обърнем внимание, че за всяка заявка освен l и r, трябва да пазим и номера ѝ в първоначалната последователност, за да извеждаме отговорите в правилния ред. Също за ускоряване можем да приложим counting sort при сортирането на заявките.

Същият алгоритъм за решаване на задачата може да се реализира и с друга структура, например map, но трябва да се провери дали ще влезе в time limit-a.