В тази задача (<https://cses.fi/problemset/task/2428>) се иска да преброим всички интервали, в които броя на различните числа е по-малък или равен на някакво си число k. Моля преди тоя документ вижте тоя за броя на различните числа в интервал или прочетете някъв туториал като тоя <https://stackoverflow.com/questions/39787455/is-it-possible-to-query-number-of-distinct-integers-in-a-range-in-olg-n> (тука гледате коментара който почва със There is a well-known **offline method** to solve this problem.)

Тази задача принципно може да се реши по доста начини, но ние ще се придържаме към този, който има доста общо със задачата за брой на различни числа в интервал. Тъй като един интервал се състои от две стойности l и r, нека фиксираме едната и се опитаме да изчислим колко възможни стойности има за другата. Нека фиксираме l. Сега задачата е просто за всяко l да смятаме кой е най-големият индекс ind, за който броя на различните числа в интервала [l, ind] е <=k. След като знаем това просто ще прибавяме ind – l + 1 към отговора, понеже всяко r, в интервалът [l, r] образува правилен сегмент според решението.

Най-лесният начин да направим това нещо е следният: да поддържаме правилно дървото и чрез байнъри сърч да намерим най-десният такъв индекс. Това ще работи, понеже колкото по-надясно отиваме, толкова по вероятно е да надхвърлим лимита k. Също така проверката ще е лесна, понеже просто пускаме куери по дървото. Това решение е O(N\*logN\*logN).

Тъй като решението не е достатъчно бързо могат да се приложат следните две бързи оптимизации като задаваме стойностите на двоичното търсене l и r, можем да сложим l = i + k – 1, тъй като със сигурност в първите k числа след i няма да имаме повече k различни такива. Също така можем да сложим r = lastInd, понеже очевидно ние няма да можем да отидем по-надясно, тъй като всички интервали, които са започвали с i+1 след тази точка са били с повече от k числа и очевидно няма шанс като тези интервали се разширяват. Принципно това е достатъчно да се реши задачата, ето го и кодът - <https://pastebin.com/YYVbDiBe>. Друга оптимизация е да използваме дърво, което е имплементирано чрез масив и да го сложим във функцията мейн, така че то да бъде stack-allocated, а не heap-allocated. Както съм казвал хиляда пъти, така то ще бъде по-често в кеша на процесора. Ето едно такова решение - <https://pastebin.com/wnRgEWsJ>. В интерес на истината, при него, ако се махнат тъпите отпимизации с границите на динамичното то успява да мине задачата за 0.87 секунди, което не е зле. Така, че работата с паметта е много важна.

Разбира се, както много разбирачи от с++ обичат да казват, никой оптимизационен гуру не може да ви спаси от лошия алгоритъм (или нещо подобно), затова ще се опитаме да напишем задачата малко по-умно. Още от миналия подход забелязахме, че можем да вземем някаква информация за „отговорът“ за всяко i от този за i+1. По-внимателни разсъждения могат да ни доведат до извода, че реално стойността на ind (максимално дясната граница, при която броя на различните числа е <=k) реално винаги ще намалява. Това е доста логично, понеже няма как някъв интервал, който се съдържа в нашия да може да се екстендне повече от нашия, който е очевидно по-голям. С други думи казано, индексът ind ще се мести заедно с i – отдясно наляво. При тази ситуация можем да използваме техниката two pointers или техника прозорец, и да нагласяме ind спрямо нашето i, по конкретно в един while цикъл ще намаляме ind, докато интервалът [i, ind] не стане с брой различни числа <=k. Taka решението пада до O(N\*logN). Ето и една примерна имплементация на това решение <https://pastebin.com/jd4VwAjN>.