АНАЛИЗ
НА РЕШЕНИЕТО НА ЗАДАЧА
РЕДИЦА

Нека разгледаме следната редица с три различни числа, т.е. K=3.

Индекс	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Стойност	2	3	3	2	4	3	4	5	3	5	3	2	4	5	4

След като отбележим редиците, които ни интересуват, се вижда следния алгоритъм:

1. Броим различните числа, и когато те станат по-големи от K, т.е. 4, отиваме в стъпка 2
2. Преброяваме числата в редицата, сравняваме с максималната дължина, отиваме в стъпка 3
3. Избираме от кой номер /индекс/ да започне новата редица, отиваме в стъпка 4.
4. Ако сме стигнали края на редицата спираме, иначе се връщаме се в стъпка 1.

Проблем са стъпки 1 и 3, от които зависи времето и използваната памет.

Индекс	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	3	2	4	3	4	5	3	5	3	2	4	5	4
2	2	3	3	2	4	3	4	5	3	5	3	2	4	5	4
3	2	3	3	2	4	3	4	5	3	5	3	2	4	5	4
4	2	3	3	2	4	3	4	5	3	5	3	2	4	5	4
5	2	3	3	2	4	3	4	5	3	5	3	2	4	5	4

При реализацията на т.1: Ако използваме масив, за да запомняме кои числа са в редицата, ще ни трябва памет за 10^9 числа. При тестове, за които стойностите на числата са от 1 до 100000, такова решение на проблема е удачно. Как да отбелязваме различните числа– стандартния начин – с 0 или 1, и поддържаме брояч на използваните различни числа.

При реализацията на т.3: В примера – в индекс 8 достигаме до числото 5, което става четвърто различно, Можем да се върнем назад и с друг брояч да отчитаме колко са различните числа: Когато този брояч стане по-голям от K, спираме. В случая това става при индекс 4. Отбелязваме, че новата редица ще започне от индекс 5 и продължаваме с индекс 9.

Индекс	1	2	3	4	5	6	7	8
Стойност	2	3	3	2	4	3	4	5
Брояч				**4**	3	3	2	1

От по-предната таблица се забелязва, че през едно число се минава най-много 3 пъти /в индекс 11 и 12 числата са оцветени 3 пъти/, защото K=3.

Ако вземем друга редица, но с K=4, ще има числа, през които се минава 4 пъти- в случая те са с индекси 8 и 9.

Индекс	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	2	3	3	3	4	4	5	5	6	6	6	7	7	8
2	2	2	3	3	3	4	4	5	5	6	6	6	7	7	8
3	2	2	3	3	3	4	4	5	5	6	6	6	7	7	8
4	2	2	3	3	3	4	4	5	5	6	6	6	7	7	8

Ясно е, че при големи K, сложността на алгоритъма се увеличава.

Затова предлагаме решение, което търси „излишното” число не отдясно-наляво, а отляво-надясно, при което през всяко число от редицата се минава 2 пъти, независимо колко е K.

За целта в *map* M, /ясно е защо не в масив/, ще записваме колко пъти се среща всяко число.
Поддържаме указател UK към най-левия елемент в **текущата редица**.
За онагледяване ще разгледаме по-подходящ пример:

Индекс		1	2	3	4	5	6	7	8
Стойност		2	3	2	3	3	4	4	5

И сега K ще е 3. Стигнали сме до индекс 8. Различните числа в редицата са 2, 3 и 4, а стойностите на съответните M са 2, 3 и 2. UK сочи към 1.

Индекс		1	2	3	4	5	6	7	8
Стойност		2	3	2	3	3	4	4	5
M[2]=	2								
M[3]=	3								
M[4]=	2								

Влизаме в цикъл:

UK=1 – намаляваме M[2] с 1 /2 е стойността на числото в позиция UK=1/

Индекс		1	2	3	4	5	6	7	8
Стойност		2	3	2	3	3	4	4	5
M[2]=	1								
M[3]=	3								
M[4]=	2								

UK=2 – намаляваме M[3] с 1 /3 е стойността на числото в позиция UK=2/

Индекс		1	2	3	4	5	6	7	8
Стойност		2	3	2	3	3	4	4	5
M[2]=	1								
M[3]=	2								
M[4]=	2								

UK=3 – намаляваме M[2] с 1 /2 е стойността на числото в позиция UK=3/

Индекс		1	2	3	4	5	6	7	8
Стойност		2	3	2	3	3	4	4	5
M[2]=	**0**								
M[3]=	2								
M[4]=	2								

M[2] стана 0 – край на цикъла.

Ясно е, че новата редица ще започне от индекс 4, затова вдигаме указателя с 1, става UK=4.
Правим M[5]=1. В структурата *map* сега имаме M[3]=2, M[4]=2 и M[5]=1.
По този алгоритъм, повече няма да се наложи да се обхождат числата с индекси от 1 до 3.
Това е така, защото докато обхождаме масива отляво-надясно, ние преместваме UK само надясно.

Автор: Павел Петров