АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА МИШКИ

Нека сме записали номерата на хранилките в масив *h*, т.е. в *h[1]* е записан номера на хранилката, от която яде мишката, която в началото живее в клетка 1, в *h[2]* – номера на хранилката, от която яде мишката, която в началото живее в клетка 2 и т.н.
Ако си представим, че на гърба на всяка мишка е залепен етикет с номера на хранилката, от която се храни, изпълнението на всяка операция по разместване на мишки е равносилна на това да разменим съдържанието на две съседни групи от по *K* елемента в масива *h*, като във всяка група подредбата на елементите се запазва. Нека тази операция наречем *K-операция*. Тогава задачата може да се преформулира така: *може ли чрез неколкократно изпълнение на K-операция върху елементите на масива h да се достигне до пермутация, в която липсват инверсии, т.е. до сортиран масив {1,2,3,…..,N}?*
Така че забравяме за мишките и работим с елементите на масива *h.*

Първото наблюдение, което трябва да направим е, че *ако първоначално h[i]=p, то след неколкократното изпълнение на K-операция по размяна на елементи, числото p може да се окаже само в елементи на масива h с индекси i±m*K*, където m е цяло, неотрицателно число, такова че въпросният индекс е валиден.

Това наблюдение позволява да представим пермутацията, която първоначално се въвежда, чрез двумерна таблица, съдържаща *K* реда и *N/K* (ако *N* се дели точно на *K*) или *N/K+ 1* (ако *N* не се дели на *K*) стълба. В ред 1 са записани стойностите на масива *h*, намиращи се в елементи *h[1], h[1+K], h[1+2*K]….h[1+(N-1)/K]*. В ред 2 са записани стойностите на масива *h[2], h[2+K], h[2+2*K],…, h[2+(N-2)/K]* и т.н. Последният ред може да не е пълен, ако *N* не се дели точно на *K*. Т.е. в един ред са елементите, чийто стойности се разменят (на два от тях) при изпълнение на *K-операция*.
Пример:
N=8, K=3
Начална пермутация: 5, 1, 6, 2, 7, 4, 8, 3
Съответна двумерна таблица:

5	2	8
1	7	3
6	4	

Ако от двумерната таблица искаме да възстановим пермутацията, то просто трябва да прочетем таблицата по стълбове.
Ако работим с двумерната таблица, една *K-операция* се изпълнява като се разменят стойностите на двойки съседни елементи от всички редове. Тези двойки са:
- от два съседни стълба, ако най-левият елемент от тези, които ще се разменят, е разположен на първия ред (на примера, ако трябва да разменим {5,1,6} с {2,7,4}, то ще разменим стойностите на елементите от първия и втория стълб);
- от три съседни стълба, ако най-левият елемент от тези, които ще се разменят,е разположен на по-долен ред (на примера, ако трябва да разменим {1,6,2} с {7,4,8}, то ще разменим стойностите на 1 и 7 от втори ред, 6 и 4 от трети ред, 2 и 8 от първи ред.

В сила е следното очевидно твърдение: *ако, прилагайки неколкократно K-операция, можем да достигнем до сортиран масив {1, 2, 3,…..,N}, то значи, че редовете на съответната двумерна таблица ще се окажат също сортирани във възходящ ред*.

Обратното не е вярно: може да сме успели да получим, прилагайки неколкократно *K-операция*, сортирани редове от двумерната таблица, но това не означава, че целият масив е сортиран.
Така че, възможността за получаване чрез *K-операции* на таблица със сортирани в нарастващ ред редове е *необходимо*, но *не достатъчно* условие за достигане до сортиран масив чрез *K-операции*.
И все пак това необходимо условие дава идея за решаване на задачата: *ако измислим алгоритъм, чрез който да се установява дали чрез K-операции може да се достигне до таблица със сортирани редове, то можем да сортираме всички редове и да проверим, прочитайки таблицата по стълбове, дали се е получил сортиран масив {1,2,3,….N}.*

Да забележим, че всяка *K-операция* променя четността на броя на инверсиите във всеки ред на таблицата, тъй като, както вече казахме, във всеки ред се разменят стойностите на два съседни елемента. Ако, чрез *K-операции*, успеем да достигнем до таблица, в която всички редове са сортирани, това значи, че броят на инверсиите във всеки ред е 0. Тогава имаме следното ***необходимо условие***: *За да можем, чрез K-операции, да достигнем до таблица, в която всички редове са сортирани във възходящ ред, трябва в таблицата, съответстваща на началната пермутация в масива h, бройките инверсии във всеки ред да са с еднаква четност.*

Ще докажем, че това условие е и достатъчно:

Най-долния (*K*-ти) ред от таблицата можем да сортираме във възходящ ред, прилагайки *K-операции* без всякакви проблеми. Просто трябва да разменяме стойностите на цели стълбове в таблицата, докато се получи сортиран последен ред. Нека сме сортирали последните *m* реда, прилагайки *K-операции*, т.е. редове с номера *K-m+1*, *K-m+2*,…, *K-1*, *K* вече са сортирани. Да се опитаме да сортираме ред с номер *K-m* като използваме *K-операции* без да развалим сортировката на долните редове.

ред *K-m*	c_1	c_2	….	c_p	c_{p+1}	c_{p+2}	.....
ред *K-m+1*	a_1	a_2	….	a_p	a_{p+1}	a_{p+2}	.....
ред *K*	b_1	b_2	….	b_p	b_{p+1}	b_{p+2}	.....

Нека в ред с номер *K-m*, първото място, в което се нарушава сортировката му, е стълб с номер *p*, т.е. $c_p > c_{p+1}$. Нека съществува и елемент c_{p+2}, т.е. елемент c_{p+1} не е последен в реда. Да приложим *K-операция*, започвайки от елемент c_p надолу, през елементите $a_p,\ldots,b_p$ и елементите от стълб *p+1*, които са над ред *K-m*. Тогава c_p и c_{p+1}, a_p и $a_{p+1},\ldots, b_p$ и b_{p+1} ще разменят стойностите си, което ще оправи инверсията в ред *K-m*, но ще породи по една инверсия във всички по-долни редове (какво става в редовете над ред с номер *K-m* не ни интересува). Да приложим още веднаж *K-операция*, но започвайки от елемент a_p надолу. Елементи a_p и $a_{p+1},\ldots, b_p$ и b_{p+1} ще разменят отново стойностите си, възстановявайки сортировката на редовете по *K-m*. Елементи c_{p+1} и c_{p+2} ще разменят стойностите си и в резултат на тези две операции ще знаем със сигурност, че $c_p < c_{p+2}$. Така, прилагайки многократно *K-операция* можем да доведем нещата в ред *K-m* до положение, при което всички елементи, без евентуално последните два, са сортирани в нарастващ ред, а редовете под *K-m* не са се променили. Сега да си спомним, че бройките инверсии в началните редове са с еднаква четност и че всяка *K-операция* променя четността на броя инверсии във всеки ред, т.е. след всяка *K-операция* бройките

инверсии във всички редове остават с една и съща четност. Тъй като всичките редове след този с номер *K-m* имат брой инверсии, равен на 0, то не е възможно последните два елемента на ред *K-m* да образуват инверсия, тъй като тогава той ще съдържа точно една инверсия, което е невъзможно. С това доказахме и достатъчността на условието.

И така, за да установим, дали с *K-операции* можем да сортираме едновременно всички редове в таблицата, трябва да изчислим броя инверсии във всеки ред и да следим дали четността на тези бройки е еднаква. Изчисляването на броя инверсии в масив с *q* елемента може да се направи със сложност q^2 при наивен подход и със сложност *q*logq* при използване на подход „разделяй и владей“, като това може да се съчетае със сортиране на масива. Може да бъде реализирано и с индексно дърво. Ако за някой ред се сменя четността на броя инверсии, то отговорът на задачата е 0 – не може да се сортира масивът чрез *K-операции*. Ако всички четности са еднакви, то с едно минаване по стълбове се установява дали се е получила желаната сортировка на целия масив и се извежда 0 или 1.

Тъй като имаме *K* реда, във всеки от които има *O(N/K)* елемента, то сложността на решението, използващо наивния алгоритъм за изчисляване на броя на инверсиите е $O(K*(N/K)^2)=O(N^2/K)$. Такова решение ще получи около 50 точки. Решението, което използва алгоритъм със сложност *q*logq* за изчисляване на броя на инверсиите ще има сложност $O(K*(N/K)*log(N/K)) = O(N*log(N/K))$ и ще получи 100 точки. Предложени са две такива решения – **mice.cpp** (с merge sort) и **mice1.cpp** (с индексно дърво).

Автори: Руско Шиков и Йордан Чапъров