**Задача D1. МОНОПАЛИНДРОМ**

1. Очевидно е, че ако N е нечетно, тогава редицата е монопалиндром, защото след обръщaнeто ѝ средният елемент остава на място и не са нужни никакви размествания – отговор 0.

2. Нека N е четно. Тогава в масив vector<int> coup[65536] построяваме по време на четене на данните мап, в който векторът coup[i] съдържа позициите (измежду 1 до N), в които числото i се среща в редицата. Позициите са естествено сортирани в нарастващ ред, заради обработката на редицата от числа от ляво надясно. Векторите останали празни не представляват интерес, защото съответното число не се среща в редицата.

Създаваме помощна функция check2(int p, int q), която по зададени позиции p и q на два еднакви елемента на редицата определя минималния брой размествания на единия от тях, след които получената редица става монопалиндром. Трудната част на решението е да се съобрази, че връщаната от функцията стойност трябва да e abs(N+1-p-q).

Сега ще разгледаме два от възможните алгоритми:

 2.1. **Квадратичен**. Разглеждаме векторите от мапа, в които имаме поне две срещания на съответното число и за всяка двойка различни позиции във вектора намираме с помощната функция минималния брой размествания lmin, при които би се получил монопалиндром от тези две позиции. Всеки път, когато намерим по малък брой размествания от запомнения в променливата gmin (инизиализирана в началото с достатъчно голяма стойност), заменяме стойността на gmin със стойността на lmin. При получаване на 0 за lmin прекратяваме изпълнението и извеждаме 0. Отговорът получаваме в gmin. Най-лош случай за този алгоритъм е, когато един елемент се среща толкова пъти, колкото всички останали в редицата.

2.2. **Субквадратичен**. За всяко от числата търсим локалния минимален брой размествания, при които се получава монопалиндром и от всички локални минимуми намираме глобалния. Имаме три възможности:

А. Ако последната позиция, в която се среща един елемент е преди средата на редицата, тогава минималният брой размествания за този елемент е при местенето му надясно до позиция, която е симетрична, спрямо средата на редицата, на позицията на най-близкия му еднакъв отляво. В такъв случай сложността на определянето на минималния брой размествания за получаването на монопалиндром с този елемент е O(1).

Б. Ако първата позиция, в която се среща един елемент е след средата на редицата, тогава минималният брой размествания за този елемент е при местенето му наляво до позиция, която е симетрична, спрямо средата на редицата, на позицията на най-близкия му еднакъв отдясно. И в този случай сложността на определянето на минималния брой размествания за получаването на монопалиндром с този елемент е O(1).

В. Ако в списъка на i има позиции, както вляво от средата на редицата, така и в дясно от средата, тогава не ни остава друга възможност освен да намерим броя на разместванията на всяко срещане на i с всяко друго негово срещане. За да не изпаднем в квадратична сложност, да направим следното:

Наблюдение. Да изберем позиция x за i и да пресмятаме броя на разместванията с елементите в позиции y, y > x, в нарастващ ред на y. Тогава получаваните стойности или само намаляват, или само растат, или отначало намаляват, а след това започват да растат.

В първия случай проверката ще продължи до достигане на най-дясната позиция, а във втория и третия може да спрем проверките при получаване на стойност по-голяма или равна на вече намерената, или 0. Това линейно търсене може да се ускори, разбира се, с вариант на двоичното търсене, но поради добрата скорост на имплементацията с линейно търсене, решението с двоично търсене не е имплементирано в авторско решение.

Кр. Манев