امیرعلی فرازمند

99522329

دو رویکرد را میتوان در نظر گرفت. یکی اینکه همه ی اعداد زوج را به سمت اول آرایه جمع کنیم و دیگری اینکه همه اعداد زوج را در آخر آرایه جمع کنیم. یکی از این 2 رویکرد به ما کمترین تعداد swap را میتوان پیدا کرد.

الگوریتمش به این صورت هست که :

1- 2 متغیر startMoves=0, j=0 را میسازیم. در یک حلقه i از 0 تا n-1 هر بار که به عدد زوج رسیدیم startMoves += (i-(j++)) میکنیم.
2- متغیرهای j=n-1, endMoves=0 را میسازیم و در یک حلقه ی i از n-1 تا 0 جلو میرویم و هر وقت به عدد زوج رسیدیم endMoves +=((j--)- i) میکنیم.
3- مینیمم startMoves و endMoves به ما مینیمم مقدار swap برای رسیدن به هدف است.(2 مرحله ی قبل در هر مرحله تعداد swap جمع میشود با مجموع آن.)

```
// Function to return minimum number of swaps
// required to segregate evens and odds
int minMovesToSegregate(int* arr, int& n)
{
    int startMoves = 0, endMoves = 0, j = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] & 1)
            startMoves += i - (j++);
    }
    j = n - 1;
    for (int i = n - 1; i >= 0; i--) {
        if (arr[i] & 1)
            endMoves += (j--) - i;
    }
    return min(startMoves, endMoves);
}
```

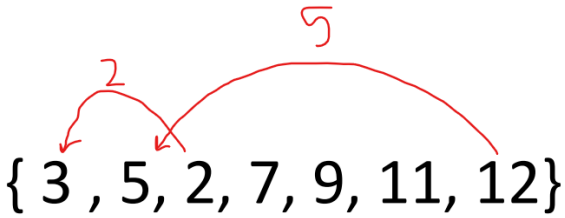
پیچیدگی زمانی :2 بار آرایه به طول n را طی میکنیم: O(n)

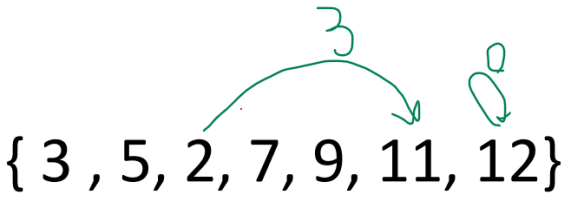
پیچیدگی حافظه :O(1)

بخش دوم:

چون 2 حالت داریم یکی اینکه همه ی اعداد زوج را به سمت اول آرایه جمع کنیم و دیگری اینکه همه اعداد زوج را در آخر آرایه جمع کنیم. یکی از این 2 رویکرد به ما کمترین تعداد swap را میتوان پیدا کرد که با مینیمم گرفتن 2 تعداد بالا میتوان به آن رسید. جواب خارج از این 2حالت نمیتواند باشد یعنی یا یمت چپ جمع میشوند اعداد زوج یا راست.

مثال سوال:





Min(7,3) = 3