اولین و ساده اندیشانه ترین راهکار موجود برای حل مسأله، آن است که از ابتدای لیست شروع به پیمایش کرده و بررسی کنیم هر عنصر چند بار در لیست تکرار شده است.

simple_find_major(A)

Require: a list $A[0, \ldots, n-1]$ containing n objects
Ensure: returns the major object whose count is more than $\lfloor \frac{n}{2} \rfloor$, -1 if no major object is found

```
for i in A do
    count = 0
    for j in A do
        if i == j then
            count += 1
    if count > len(A) // 2 then
        return i
return -1
```

همانطور که مشخص است، به ازای هر عضو i در A ، یک بار پیمایش صورت می گیرد و تعداد تکرار i در لیست A مشخص شده، در متغیر $count$ ذخیره می گردد. در انتهای پیمایش و قبل از پرداختن به عضو بعدی $(i+1)$ مقدار متغیر $count$ بررسی شده و در صورت بیشتر بودن از $len(A)//2$ مقدار i به عنوان پاسخ مسئله خروجی داده می شود. در انتهای پیمایش اعضای A اگر تا آن زمان مقداری به عنوان خروجی مشخص نشده باشد، -1 به عنوان خروجی داده می شود.

به ازای هر یک از اعضای لیست، تمام اعضای لیست یک بار پیمایش می شوند. از آنجایی که عمل پایه در حلقه اول و دوم، یک واحد مقایسه است، کارایی زمانی آن $\Theta(n \times n) = \Theta(n^2)$ خواهد بود.

کارای فضایی نیز، به دلیل وجود صرفا ۳ متغیر و عدم استفاده از لیستی جداگانه برای یافتن جواب مسأله برابر با $\Theta(1)$ خواهد بود.