برای حل این سوال به یک درخت AVL نیاز خواهیم داشت با این تفاوت که هر گره، علاوه بر داشتن اشاره‌گر هایی به گره پدر و گره فرزند راست و گره فرزند چپ، مقداری تحت عنوان $Size$ را نیز نگهداری می کند. $Size$ گرهِ c تعداد تمام گره هایی است که از c منشعب می‌شوند. برای مثال $Size$ ریشه درخت، برابر با یک واحد کمتر از کل تعداد گره هاست.

از طرفی، فرض کنید دنبال k امین‌کوچکترین عنصر در درخت هستیم. می‌دانیم تمام عناصر کوچکتر از ریشه، در سمت چپ آن قرار دارند. به عبارت دیگر، اگر $Size$ زیردرخت چپ برابر با $k-1$ باشد (یعنی $k-1$ عنصر در زیردرخت چپ قرار دارند) و به دنبال k امین‌کوچکترین عنصر باشیم، پاسخ همان ریشه است. چرا که $k-1$ عنصر کوچکتر از ریشه وجود دارند و خود ریشه k امین عنصر است.

حال اگر کلید k کوچکتر از $Size$ زیردرخت چپ باشد، می‌توان نتیجه گرفت که عنصر مورد نظر ما جایی در زیردرخت چپ قرار دارد. در غیر این صورت، اگر k از $Size$ زیردرخت چپ بیشتر باشد، باید در زیردرخت راست دنبال آن بگردیم.

میانه‌ی یک لیست که تعداد عناصر آن فرد باشد، اندیسِ $\left[\frac{N+1}{2}\right]$ ام آن است. در صورتی که تعداد عناصر آن زوج باشد، میانگین دو عنصر $\left[\frac{N}{2}\right]$ ام و $\left[\frac{N+2}{2}\right]$ ام آن است.

پس با استفاده از الگوریتمی که در بالا شرح داده شد، به یافتن عناصر مورد نظر می‌پردازیم و میانه را حساب می‌کنیم.

Find_Kth($T, K, root$)

Require: T: A tree, K: The K_th element to be found, root: The root to begin with
Ensure: The K_th smallest element in T

```
size = root.left.size + 1
if k == size then
    return root.value
else if k < size then
    return Find_Kth(T, K, root.left)
else
    return Find_Kth(T, K - size, root.right)
```

Find_Median(T)

Require: T: The AVL tree
Ensure: Median in T

```
size = T.size
if size%2 == 1 then
    return T.Find_Kth((T, size + 1, T.root) // 2)
first_med = T.Find_Kth(T, size // 2, T.root)
second_med = T.Find_Kth(T, (size + 2) // 2, T.root)
return (first_med + second_med) / 2
```

با توجه به اینکه درختی که استفاده کردیم از نوع AVL بوده است، کارایی زمانی افزودن یک مقدار به آن از مرتبه $O(\log n)$ خواهد بود. بعلاوه، همانطور که در الگوریتم $Find_Median$ مشاهده کردیم، هر بار درخت را به دو نیم تقسیم کرده و یک نیم را حذف می‌کنیم. در نتیجه کارایی یافتن میانه از مرتبه $O(\log n)$ خواهد بود.