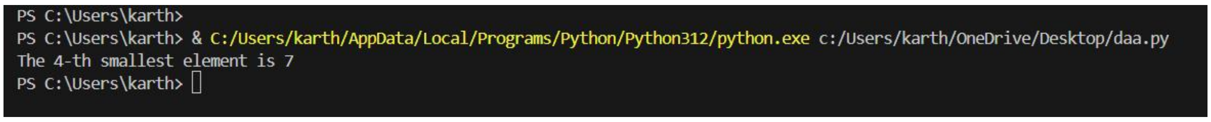
8. Median of medians.

Code:

```
def partition(arr, low, high, pivot_index):
    pivot_value = arr[pivot_index]
    arr[pivot_index], arr[high] = arr[high], arr[pivot_index]
    store_index = low
    for i in range(low, high):
        if arr[i] < pivot_value:
            arr[store_index], arr[i] = arr[i], arr[store_index]
            store_index += 1
    arr[store_index], arr[high] = arr[high], arr[store_index]
    return store_index
def select(arr, low, high, k):
    while True:
        if low == high:
            return arr[low]
        pivot_index = median_of_medians(arr, low, high)
        pivot_index = partition(arr, low, high, pivot_index)
        if k == pivot_index:
            return arr[k]
        elif k < pivot_index:
            high = pivot_index - 1
        else:
            low = pivot_index + 1
def median_of_medians(arr, low, high):
    n = high - low + 1
    if n <= 5:
        return partition5(arr, low, high)
    for i in range(0, n // 5):
        sub_left = low + i * 5
        sub_right = sub_left + 4
        if sub_right > high:
            sub_right = high
        median5 = partition5(arr, sub_left, sub_right)
        arr[low + i], arr[median5] = arr[median5], arr[low + i]
    mid = (n // 10) + low + 1
    return select(arr, low, low + n // 5 - 1, mid)
def partition5(arr, low, high):
    sublist = arr[low:high+1]
    sublist.sort()
    arr[low:high+1] = sublist
    return (low + high) // 2
arr = [12, 3, 5, 7, 4, 19, 26]
k = 3
result = select(arr, 0, len(arr) - 1, k)
print(f"The {k+1}-th smallest element is {result}")
```

output:



```
PS C:\Users\karth>
PS C:\Users\karth> & C:/Users/karth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/karth/OneDrive/Desktop/daa.py
The 4-th smallest element is 7
PS C:\Users\karth> 
```

Time complexity:

F(n)=o(n)